Migration Guide: Amazon Aurora to HeatWave MySQL on Oracle Cloud Infrastructure (OCI)

Before you start:

- You must have an account on Oracle Cloud Infrastructure (OCI) and Amazon Web Services (AWS).
- Some OCI knowledge is preferred.
- This migration guide only covers how to migrate your database from Amazon Aurora MySQL to HeatWave MySQL on OCI. Before performing the migration, you should have considered downtime (the length of the downtime will mostly depend on the size of your database and checks you may want to perform before bringing your database back online), application compatibility, current database metrics (CPU, storage size, RAM, max number of concurrent users, backups, binary logs expiration, number of replicas if any, etc.), desired database metrics, networking, security, user testing, etc.
- The migration method shown in this guide works for Amazon Aurora MySQL v5.7 and above.
- When following the guide, you should always execute the commands/steps shown as an admin/root user wherever applicable.
    - On OCI and AWS you must have the ability to create and manage resources.
    - For your Amazon Aurora MySQL instance, use an admin/root user.
- You do not need to make any configuration changes to your Amazon Aurora MySQL for this migration.
- If you have MySQL replication configured in your current Amazon Aurora MySQL environment, you can perform the migration steps shown in this guide from either your source or replica instance.
- The Overview section of this downtime migration guide contains all the steps that are needed to finish the database migration from Amazon Aurora MySQL to HeatWave MySQL on OCI.
- In the Walkthrough section of this migration guide, we will apply the information provided in the Overview section and give you a simple step-by-step guide. In this step-by-step guide, we will have an Amazon Aurora MySQL instance with some sample data pre-loaded and will migrate it over to HeatWave MySQL on OCI. This will help you follow and better visualize the process/information provided in the Overview section.
- You can use the Walkthrough section's step-by-step guide as a reference for your migration from Amazon Aurora MySQL to HeatWave MySQL. When following the guide, make changes along the way to your AWS and OCI environment accordingly or as required. Since each user following the step-by-step guide will have their environments configured differently, we cannot provide an ideal example that works for everyone.

Overview:

Following are the required steps to migrate data from Amazon Aurora MySQL to HeatWave MySQL on OCI:

I) Have an Oracle Cloud Infrastructure (OCI) account and Amazon Web Services (AWS) account.

OCI Sign in/Sign up page: https://cloud.oracle.com
AWS Sign in/Sign up page: https://aws.amazon.com/

II) Set up a VPN connection from OCI to AWS.

[A VPN connection will allow you to bridge your AWS network with the OCI VCN. The VPN connection will allow your Amazon Aurora MySQL to connect to HeatWave MySQL on OCI and it also ensures that your data in transit while it is being migrated is encrypted.]
VPN Connection to AWS: https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/vpn_to_aws.htm

III) On OCI, create a HeatWave MySQL instance.

[You can create either a Standalone or High Availability HeatWave MySQL instance. Both options are fully-managed.]
Provision OCI HeatWave MySQL: https://docs.oracle.com/en-us/iaas/mysql-database/doc/creating-db-system1.html

IV) Install MySQL Shell 8.1 or above on an EC2 instance that can connect to Amazon Aurora MySQL.

[MySQL Shell on EC2 will be used to copy DDL and data from Amazon Aurora MySQL to HeatWave MySQL on OCI. You must download MySQL Shell 8.1 or above.]
Download MySQL Shell: https://dev.mysql.com/downloads/shell/
Install MySQL Shell: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html

V) Connect to Amazon Aurora MySQL using MySQL Shell on EC2. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from Amazon Aurora MySQL to HeatWave MySQL on OCI.

[The dump created by MySQL Shell's instance copy utility comprises DDL files specifying the schema structure, and tab-separated `.tsv` files containing the data.]
MySQL Shell Copy Utilities: https://dev.mysql.com/doc/mysql-shell/8.1/en/mysql-shell-utils-copy.html

VI) (Optional) On OCI, use the Cloud Shell to verify whether the data was migrated successfully from Amazon Aurora MySQL to HeatWave MySQL on OCI.

[Cloud Shell is a web browser-based terminal accessible from the Oracle Cloud Console.]
OCI Cloud Shell: https://docs.oracle.com/en-us/iaas/Content/API/Concepts/cloudshellintro.htm

VII) (Optional) On OCI, if the HeatWave option was enabled during HeatWave MySQL DB creation, add the HW Cluster and load data from MySQL InnoDB storage into the HW Cluster using automation.

[Attaching the HeatWave in-memory Cluster combines transactions, analytics, and machine learning services into one MySQL Database.]
Add a HeatWave Cluster: https://docs.oracle.com/en-us/iaas/mysql-database/doc/adding-heatwave-cluster.html#GUID-2335AC1F-FB01-4701-9EFD-810A3489A850
Load Data into HeatWave: https://dev.mysql.com/doc/heatwave/en/mys-hw-auto-parallel-load.html

Walkthrough:

I) Have an Oracle Cloud Infrastructure (OCI) account and Amazon Web Services (AWS) account.

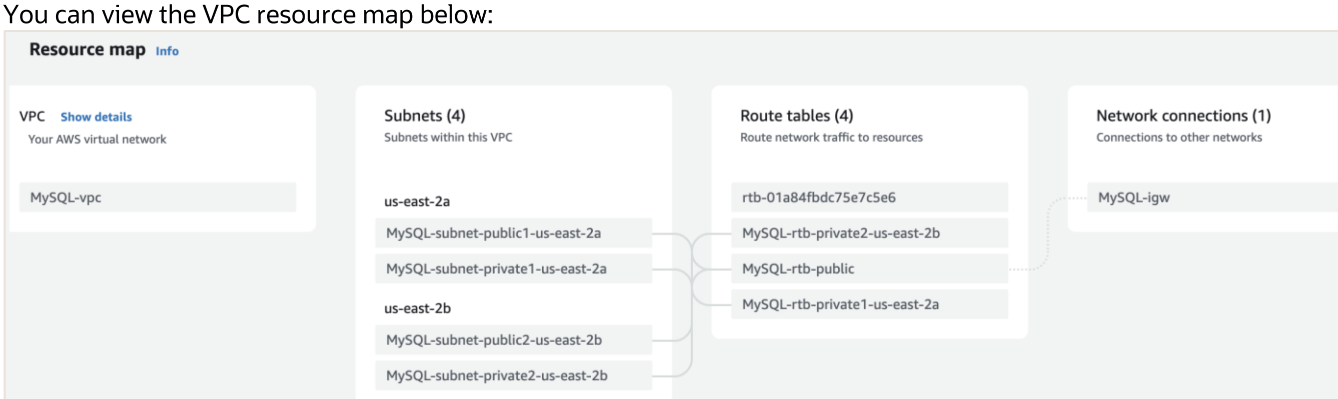
OCI Sign in/Sign up page: https://cloud.oracle.com
AWS Sign in/Sign up page: https://aws.amazon.com/

II) Set up a VPN connection from OCI to AWS.

1. Below is the Amazon Aurora MySQL instance version and [the sample database ("world")](#) that will be migrated for this guide. The sample world database consists of 3 tables. The Amazon Aurora MySQL instance used for this does not have public access.

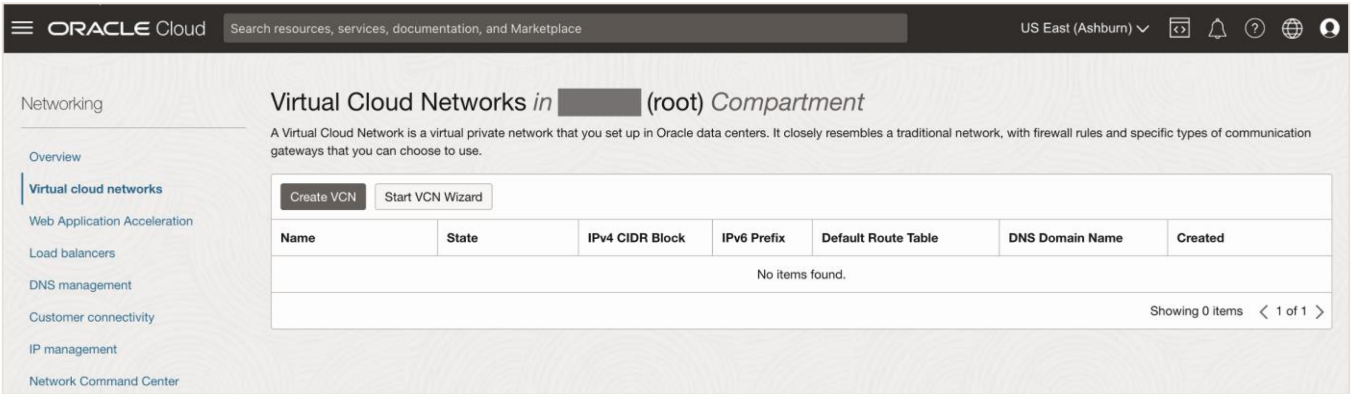
```
 MySQL  database-1-instance-1.[redacted]  world  SQL > SELECT @@VERSION;
+-----------+
| @@VERSION |
+-----------+
| 5.7.12    |
+-----------+
1 row in set (0.0025 sec)
 MySQL  database-1-instance-1.[redacted]  world  SQL > SHOW SCHEMAS;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| world              |
+--------------------+
5 rows in set (0.0010 sec)
 MySQL  database-1-instance-1.[redacted]  world  SQL > SHOW TABLES IN world;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
```

2. The AWS VPC associated with the above Amazon Aurora MySQL instance uses an IPv4 CIDR: 10.1.0.0/16. You can view the VPC resource map below:

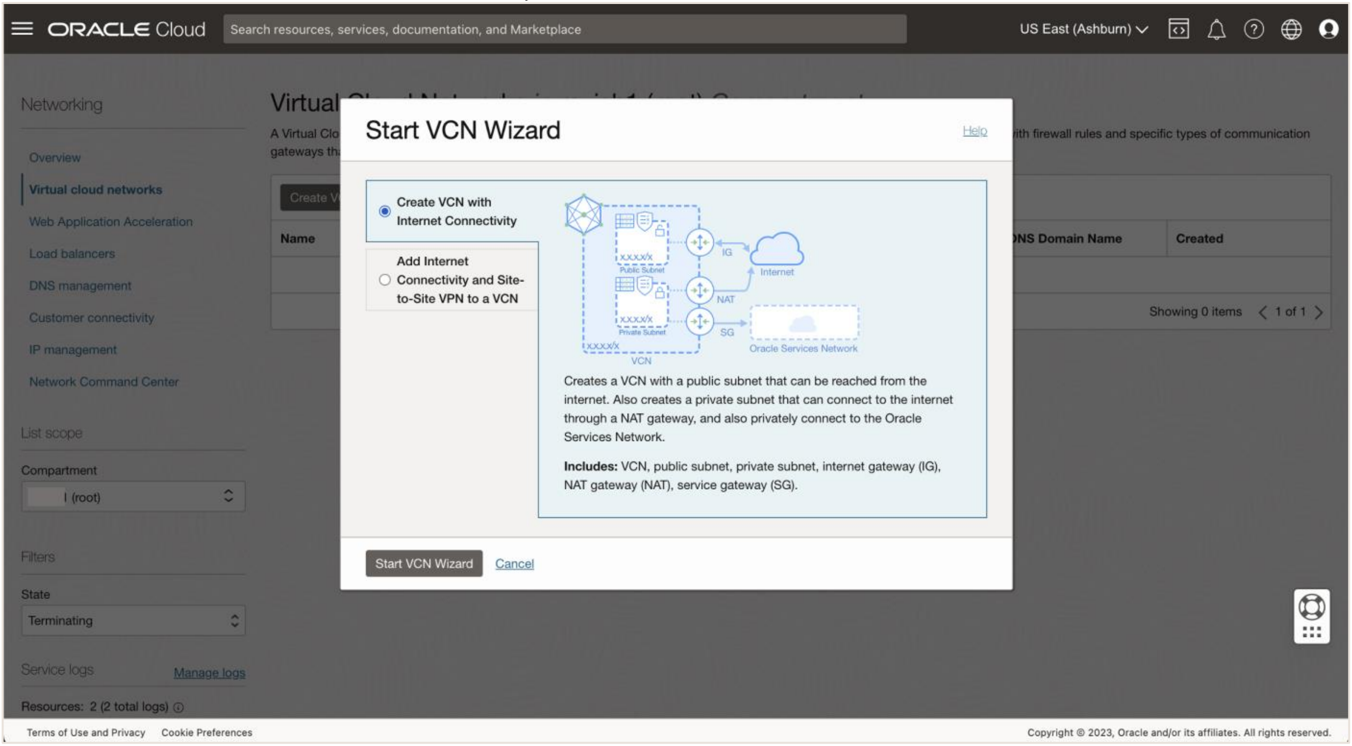


3. Log in to [OCI](#) and create a VCN. Open the OCI navigation menu, click **Networking**, and click **Virtual cloud networks**.

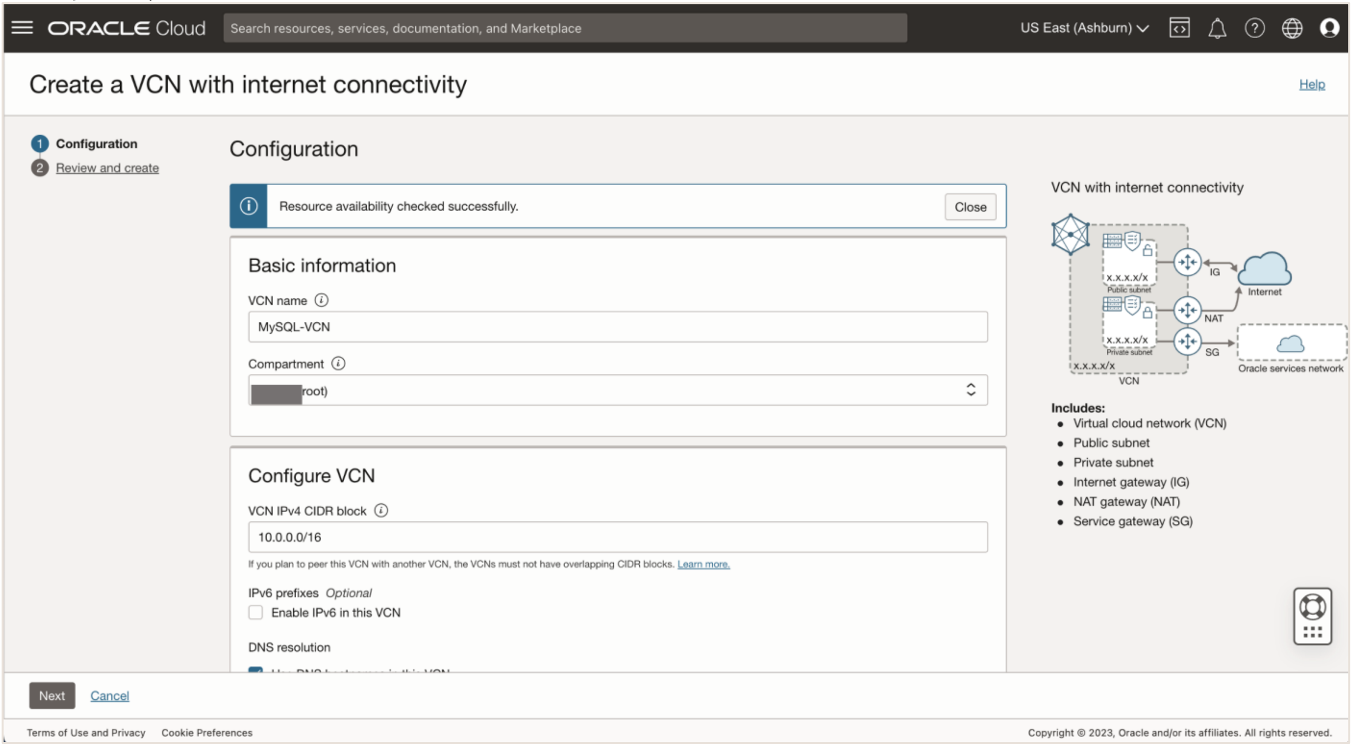
4. Ensure you are in your desired compartment - we have chosen the `root` compartment. Click **Start VCN Wizard**.



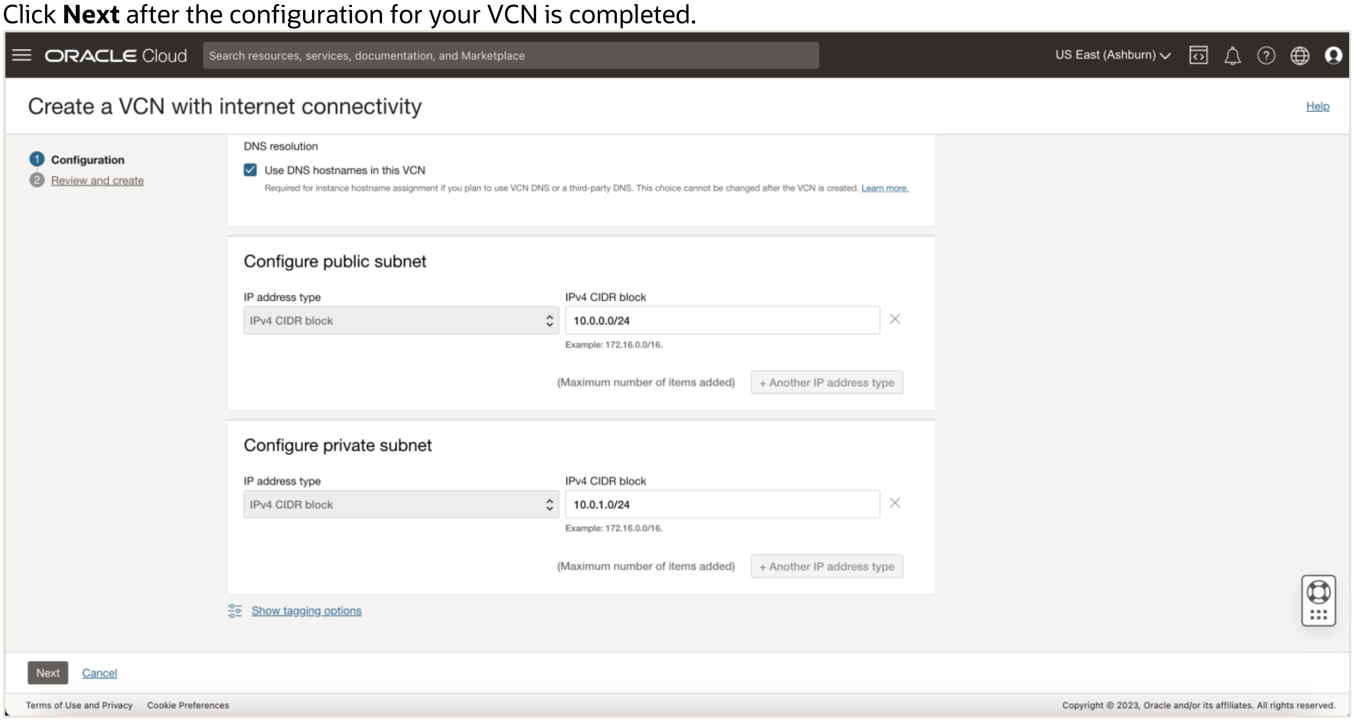
5. Select **Create VCN with Internet Connectivity** and click **Start VCN Wizard**.



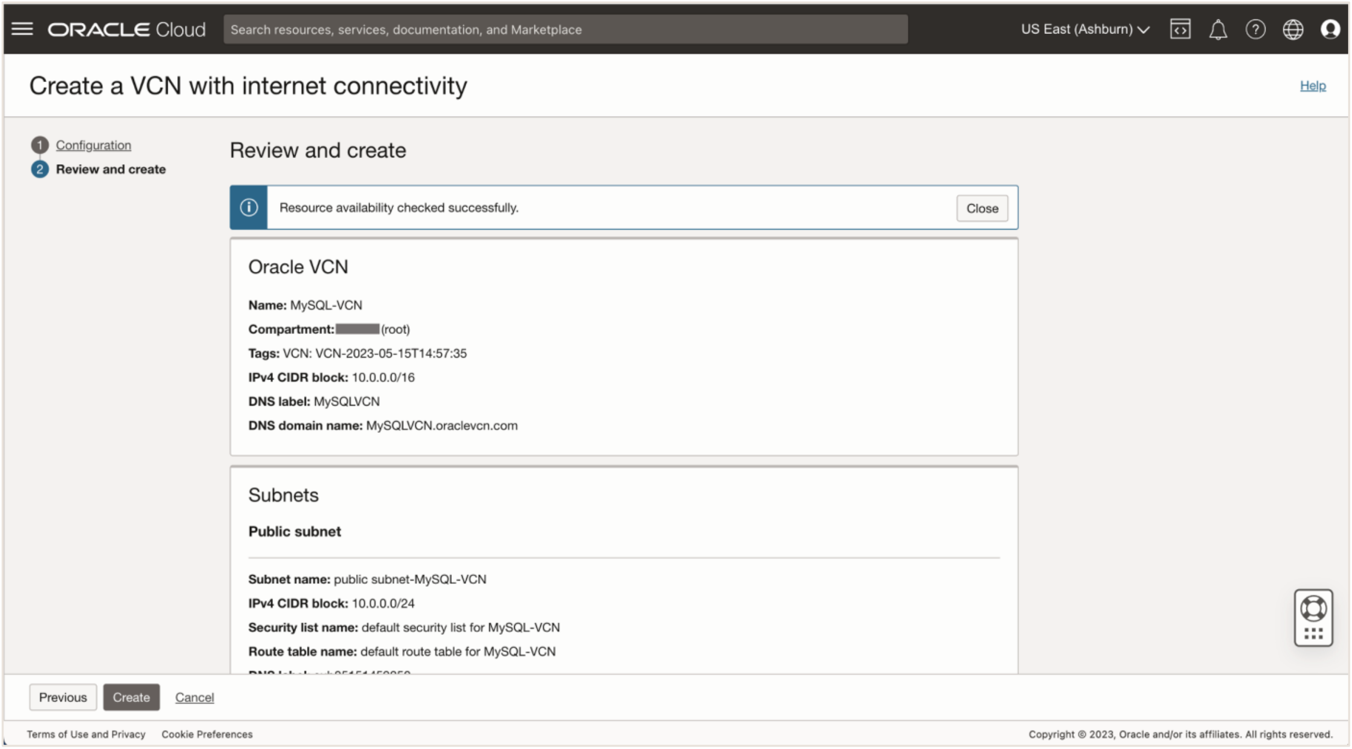
6. Enter a **VCN name** and **configure your VCN's IPv4 CIDR block** - **including the public and the private subnet**. The guide uses the default values for all. Make sure that the OCI VCN IPv4 CIDR block does not overlap with your AWS VPC IPv4 CIDR.



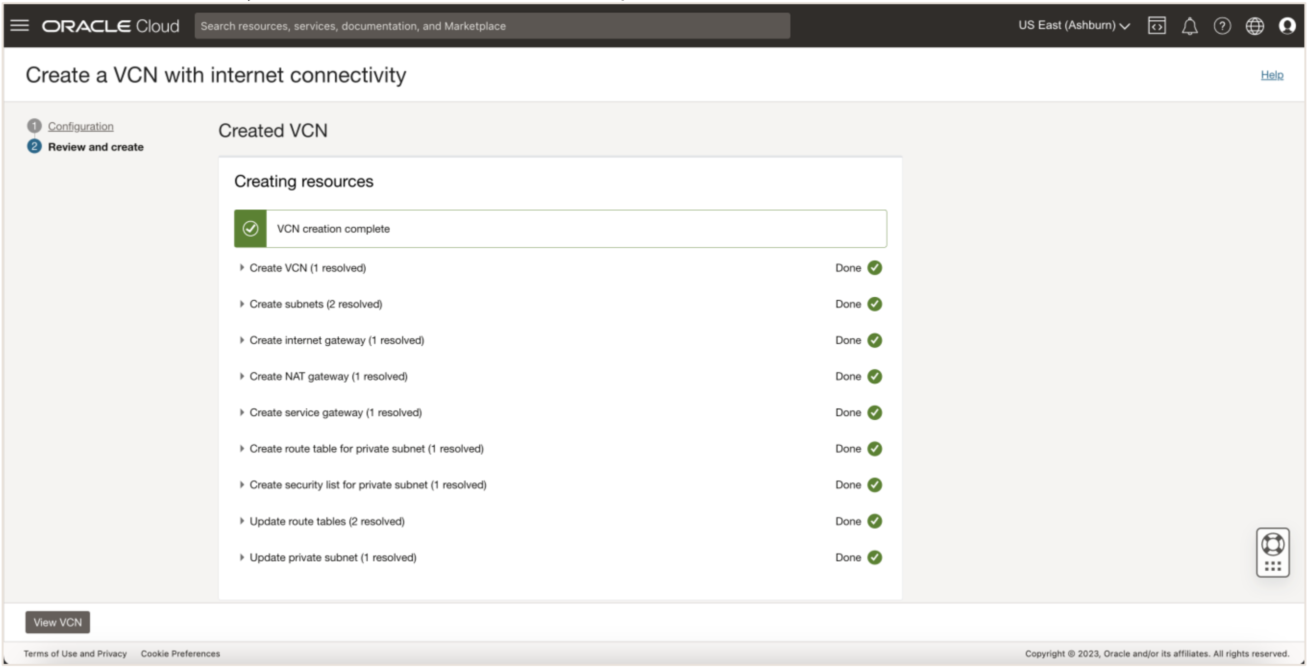
7. Click **Next** after the configuration for your VCN is completed.



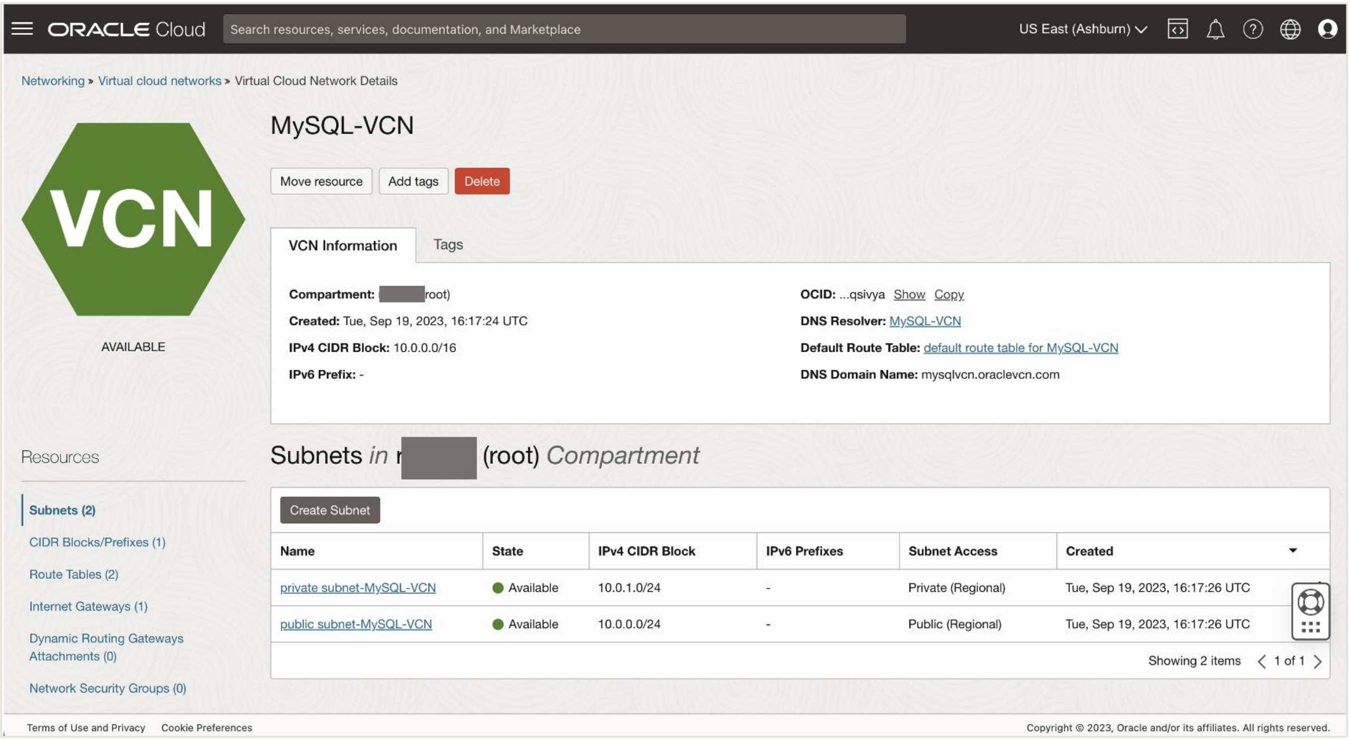
8. On the Review and create page, validate the information for your VCN and click **Create**.



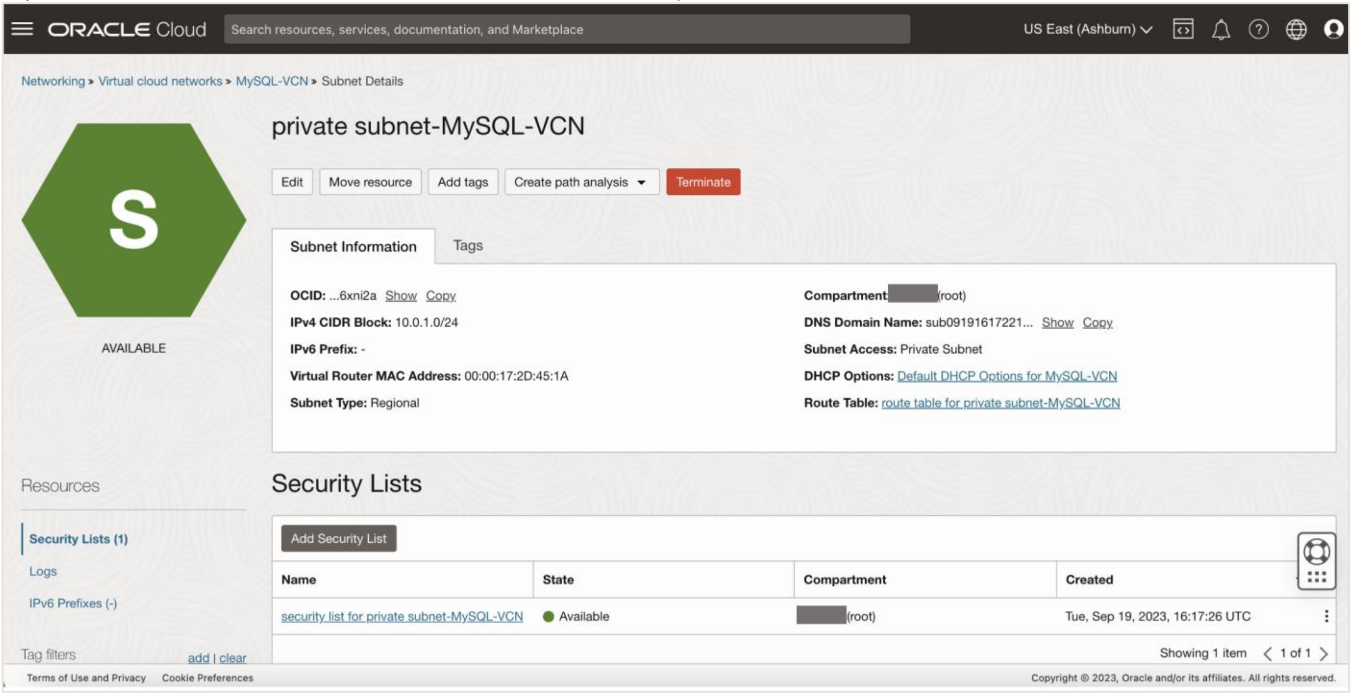
9. Click **View VCN** after your VCN creation has been completed.



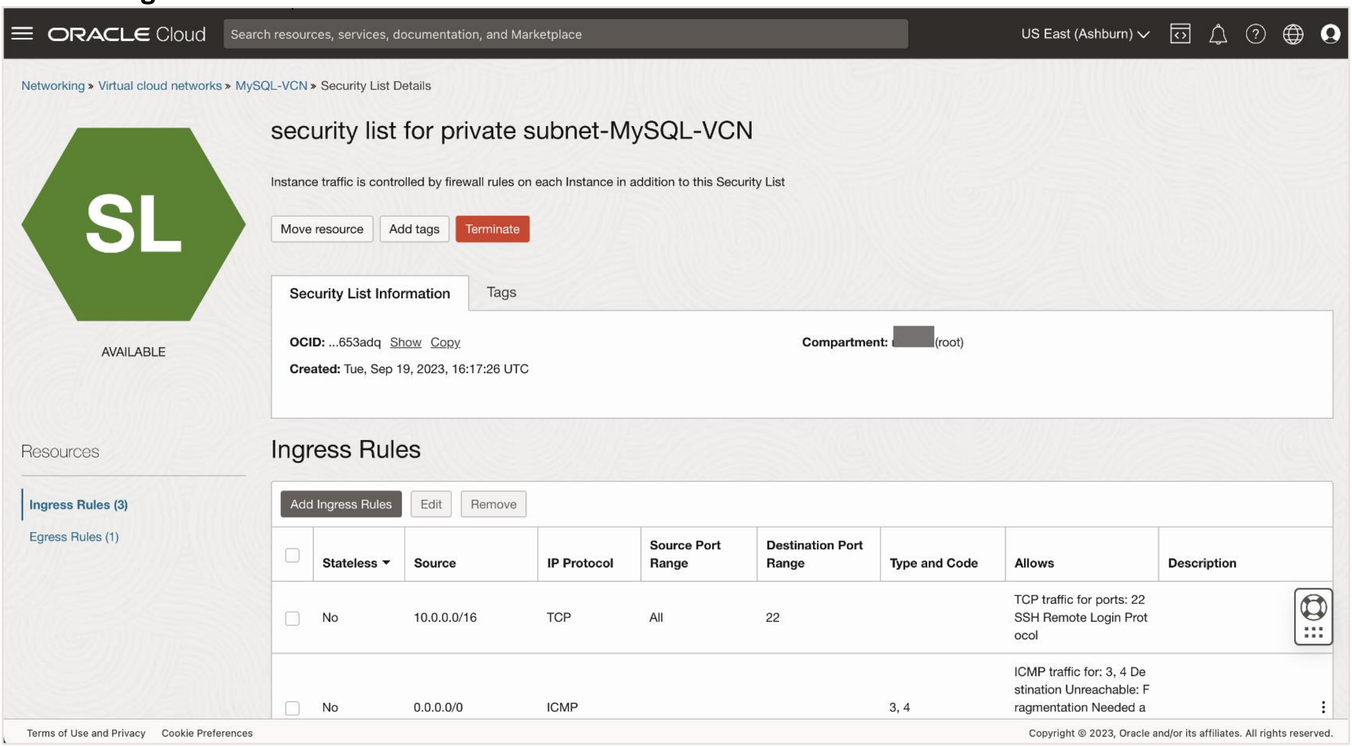
10. On the Virtual Cloud Network Details page under Resources, click **Subnets** section. Click on **private subnet-<vcn-name>**.



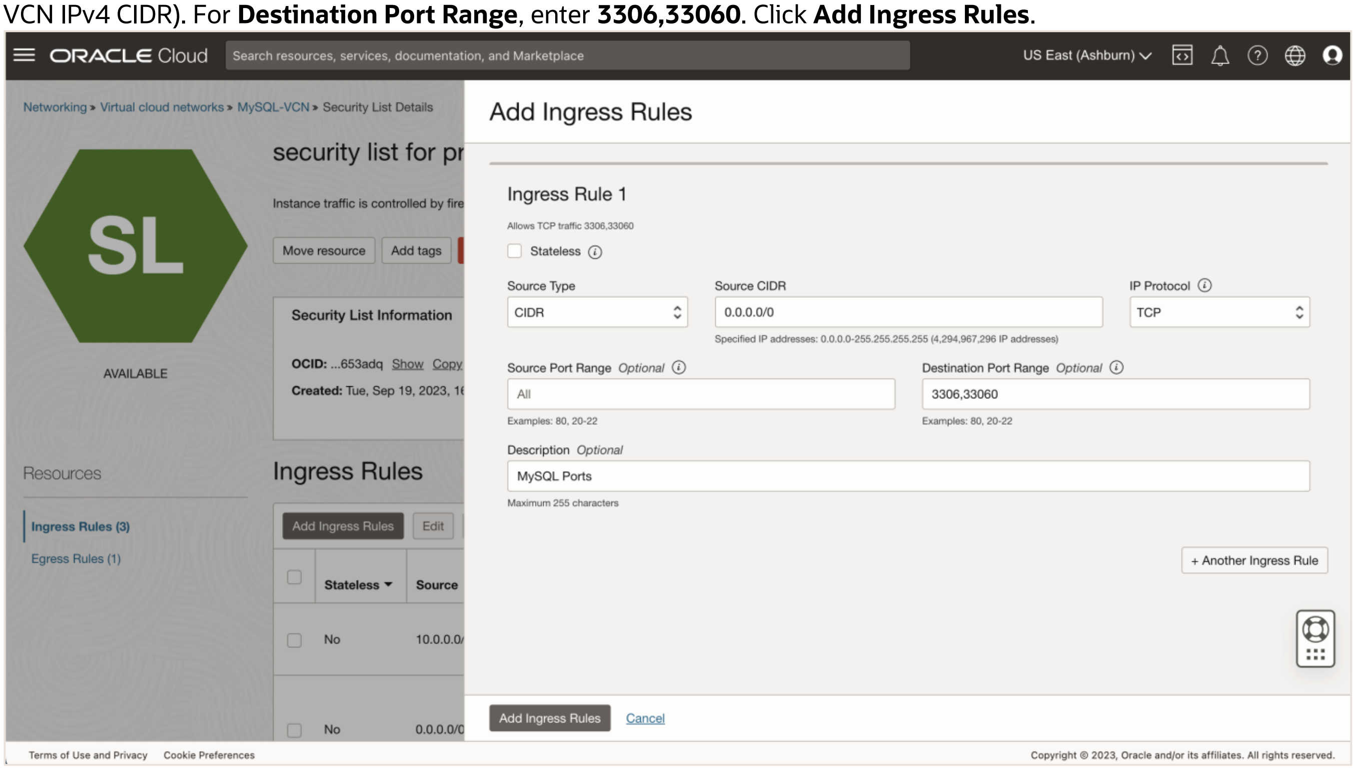
11. Click on **security list for private subnet-<vcn-name>** to add an Ingress Rule which will allow HeatWave MySQL to access the Aurora instance on AWS and the Compute instance on OCI.



12. Click **Add Ingress Rules**.

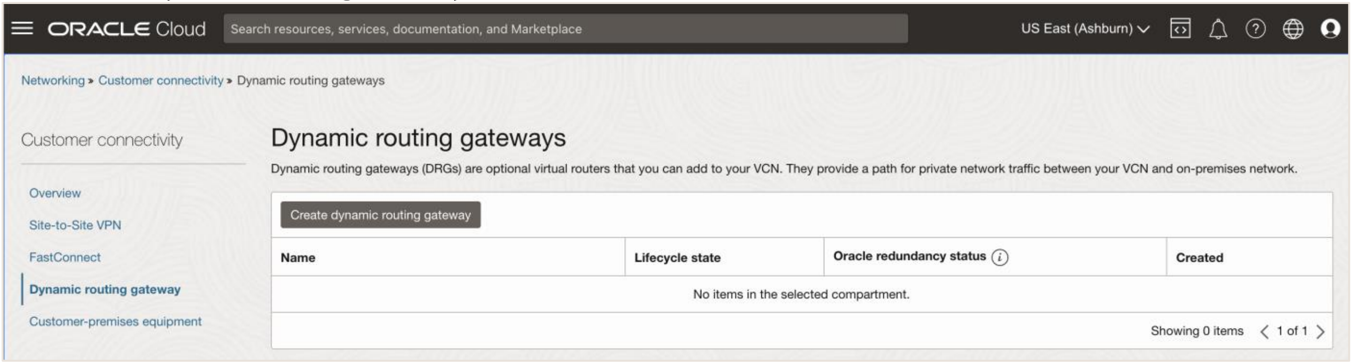


13. For **Source CIDR** type **0.0.0.0/0** (you can be more restrictive here and enter only the AWS and OCI VPC and VCN IPv4 CIDR). For **Destination Port Range**, enter **3306,33060**. Click **Add Ingress Rules**.

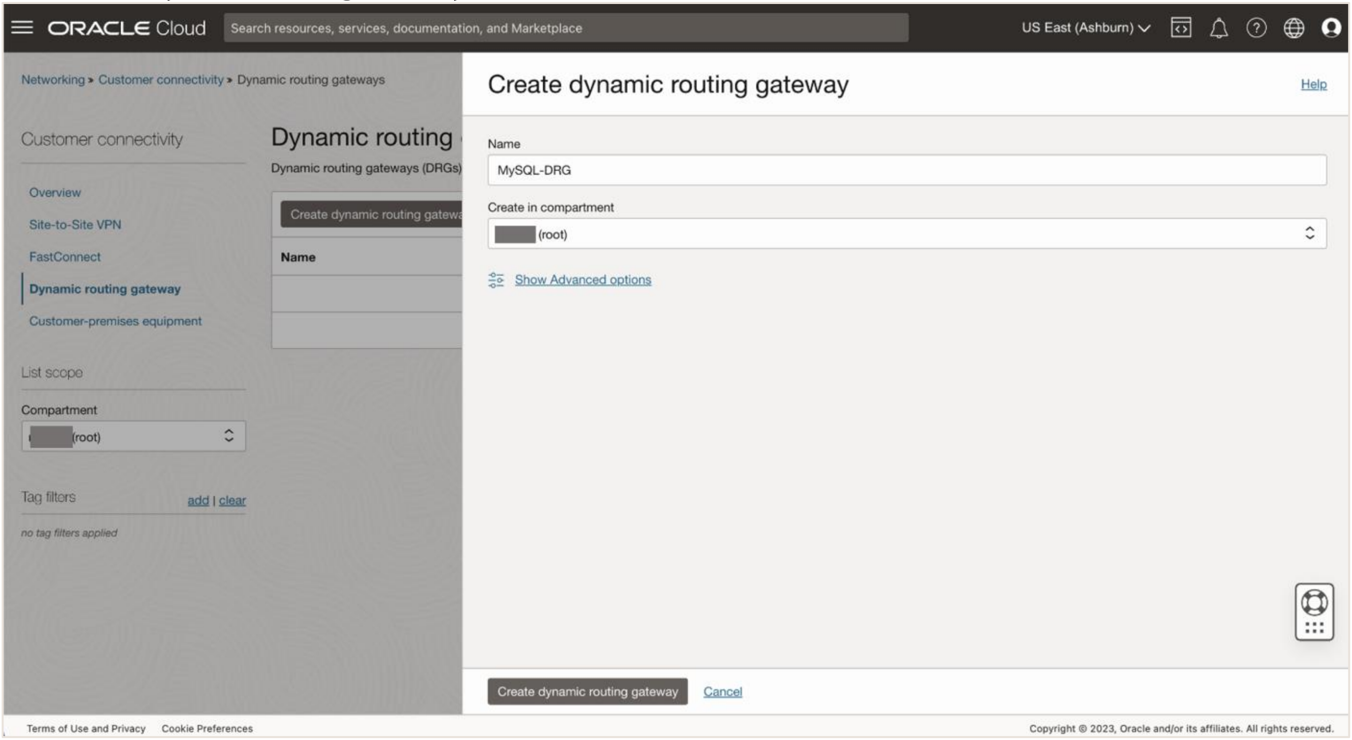


14. Open the OCI navigation menu, click **Networking** and click **Dynamic routing gateway** under Customer Connectivity.

15. Click **Create Dynamic Routing Gateway**.

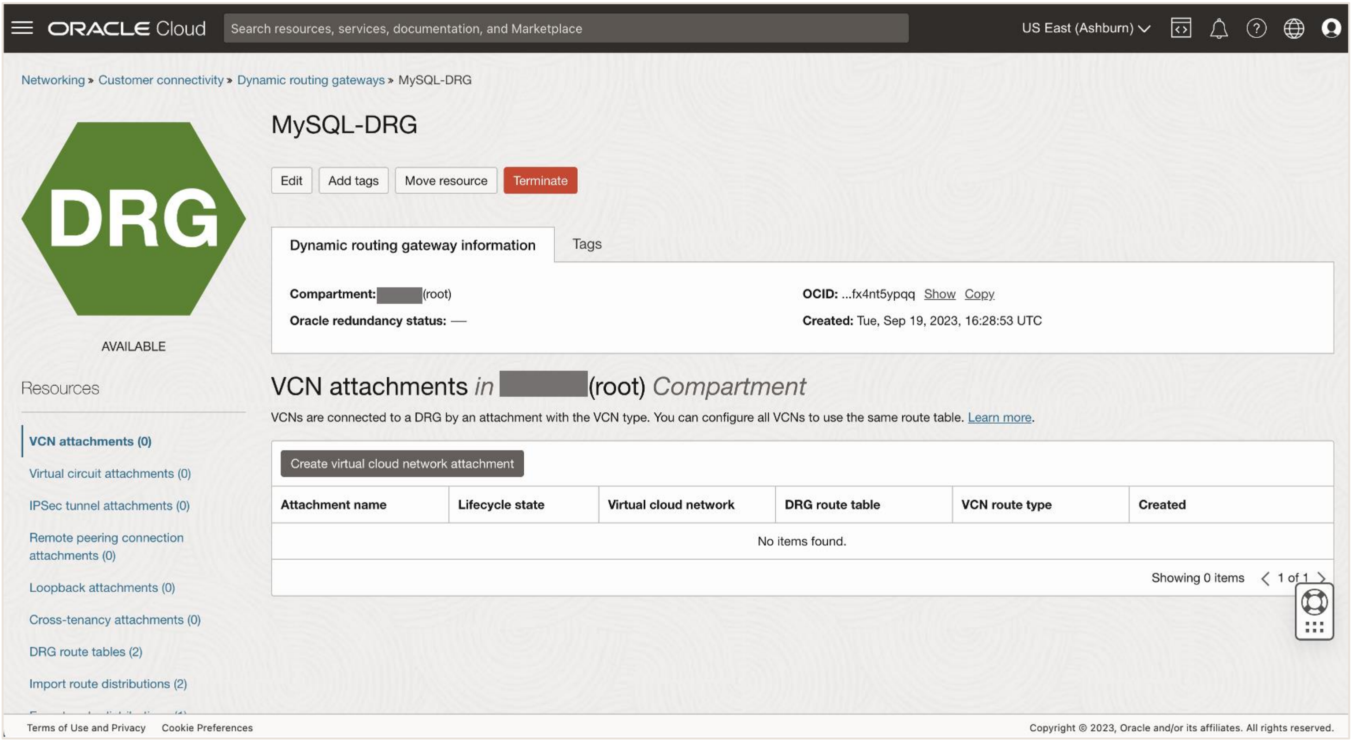


16. Enter a **DRG name**. Under **Create in compartment** - **choose the compartment where your VCN resides**. Click **Create Dynamic Routing Gateway**.

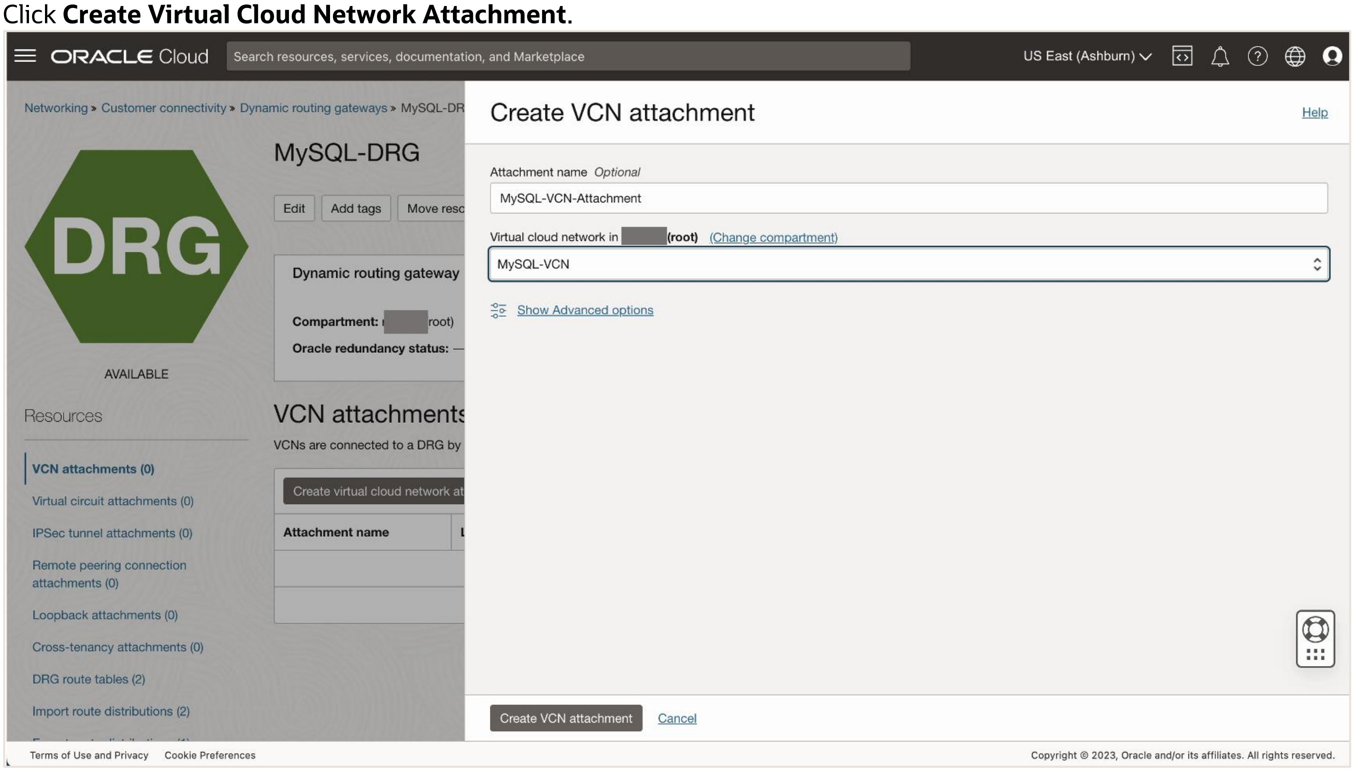


ORACLE

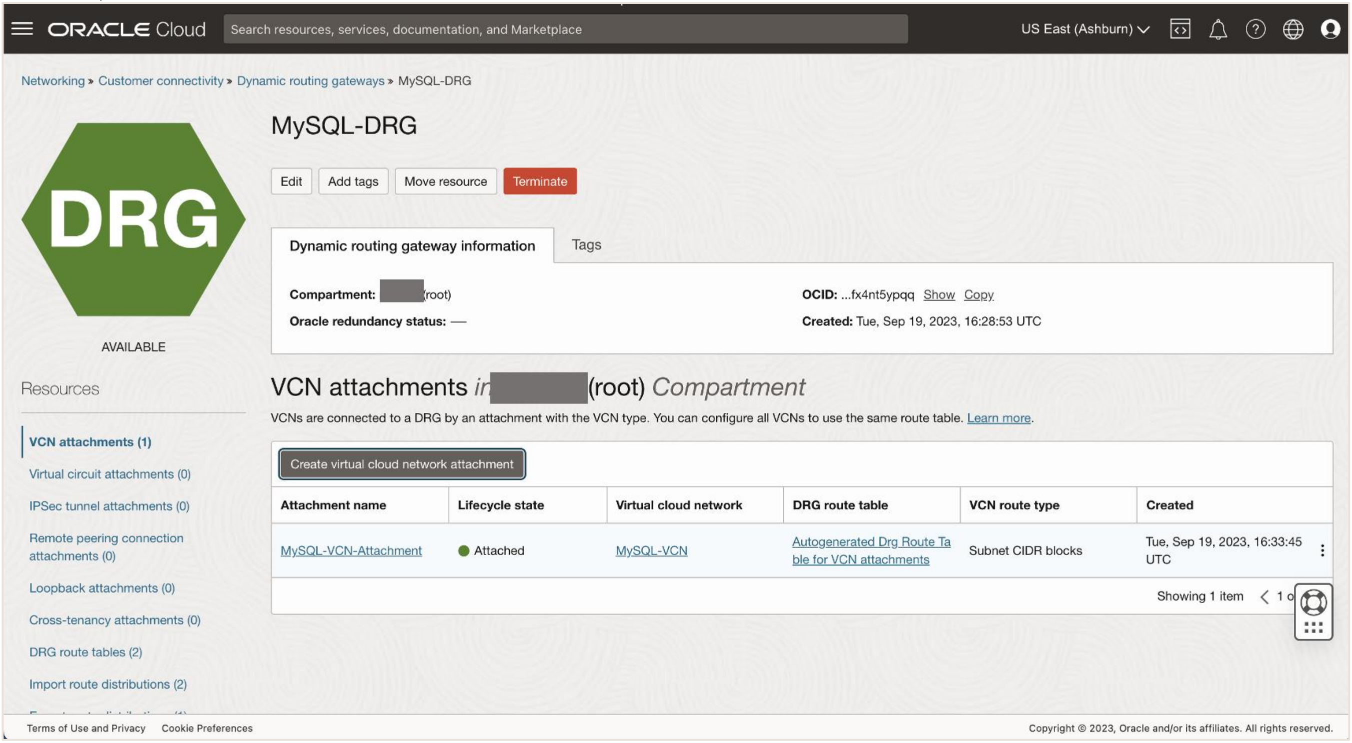
17. You will be taken to the DRG Details page. Once your DRG changes its state from Provisioning to **Available**, under Resources, click **Virtual Cloud Network Attachment**. Click **Create Virtual Cloud Network Attachment**.



18. Enter a **Virtual Cloud Network Attachment name** and **select the appropriate VCN** from the drop-down list. Click **Create Virtual Cloud Network Attachment**.

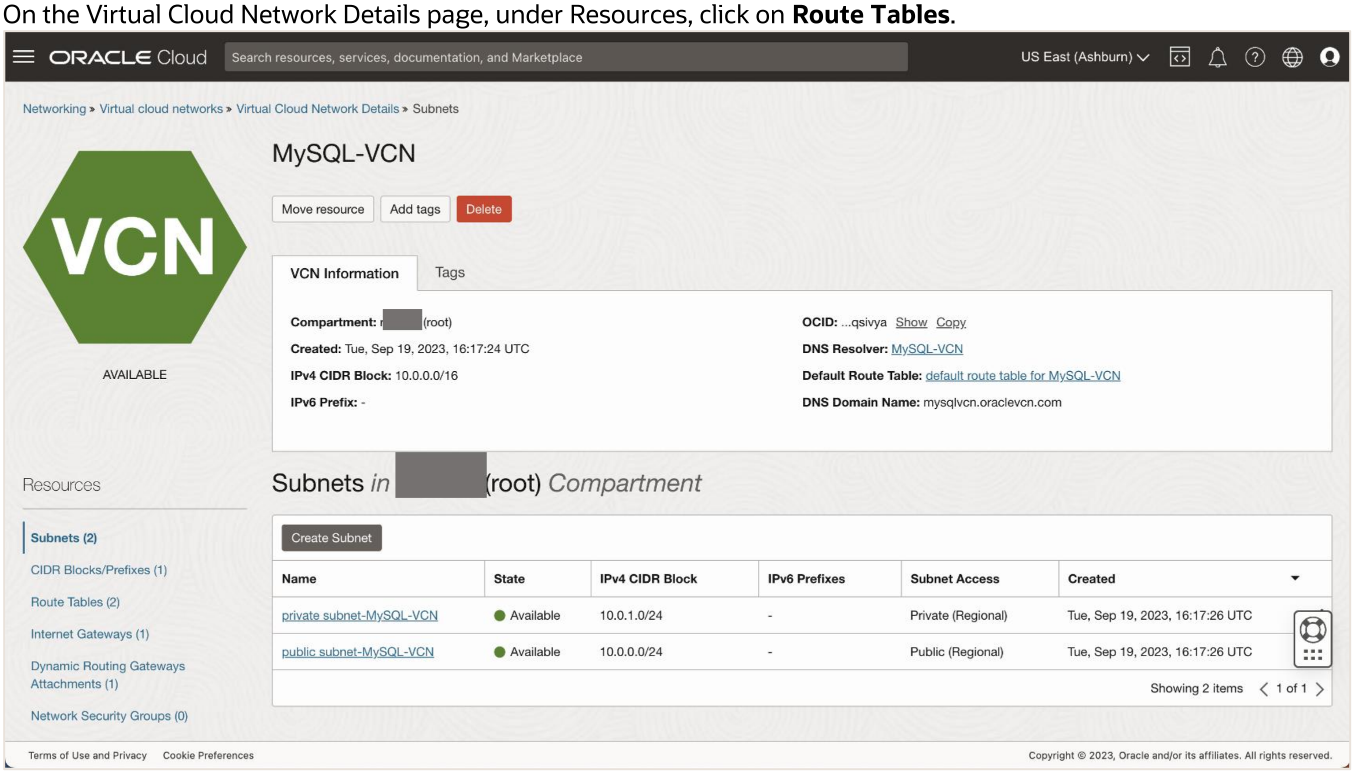


19. Wait for your VCN Attachment to be in an **Attached** state.



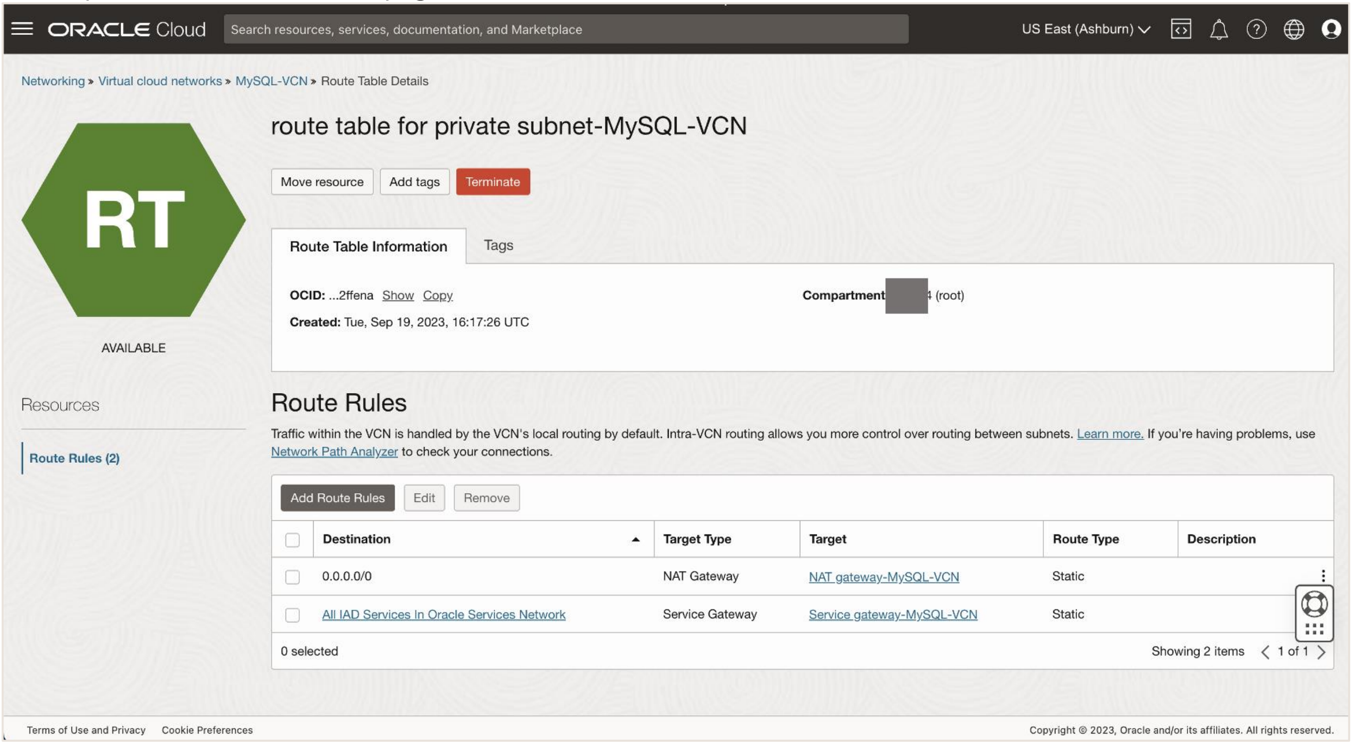
20. Open the OCI navigation menu, click **Networking** and click on **Virtual cloud networks**. After landing on the Virtual Cloud Networks page, click on **the name of your VCN**.

21. On the Virtual Cloud Network Details page, under Resources, click on **Route Tables**.

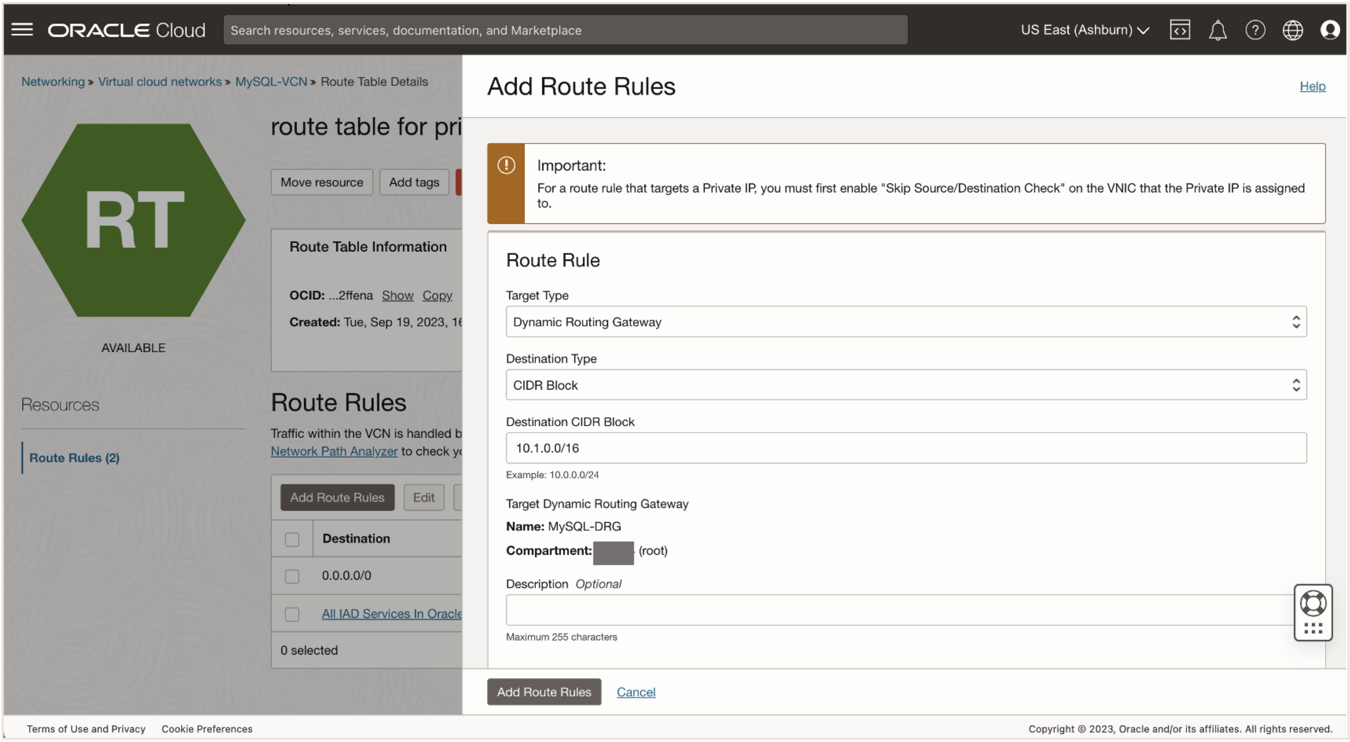


22. You should see two Route Tables, one for your private subnet and the other for your public subnet. Click on **route table for private subnet-<vcn-name>**.

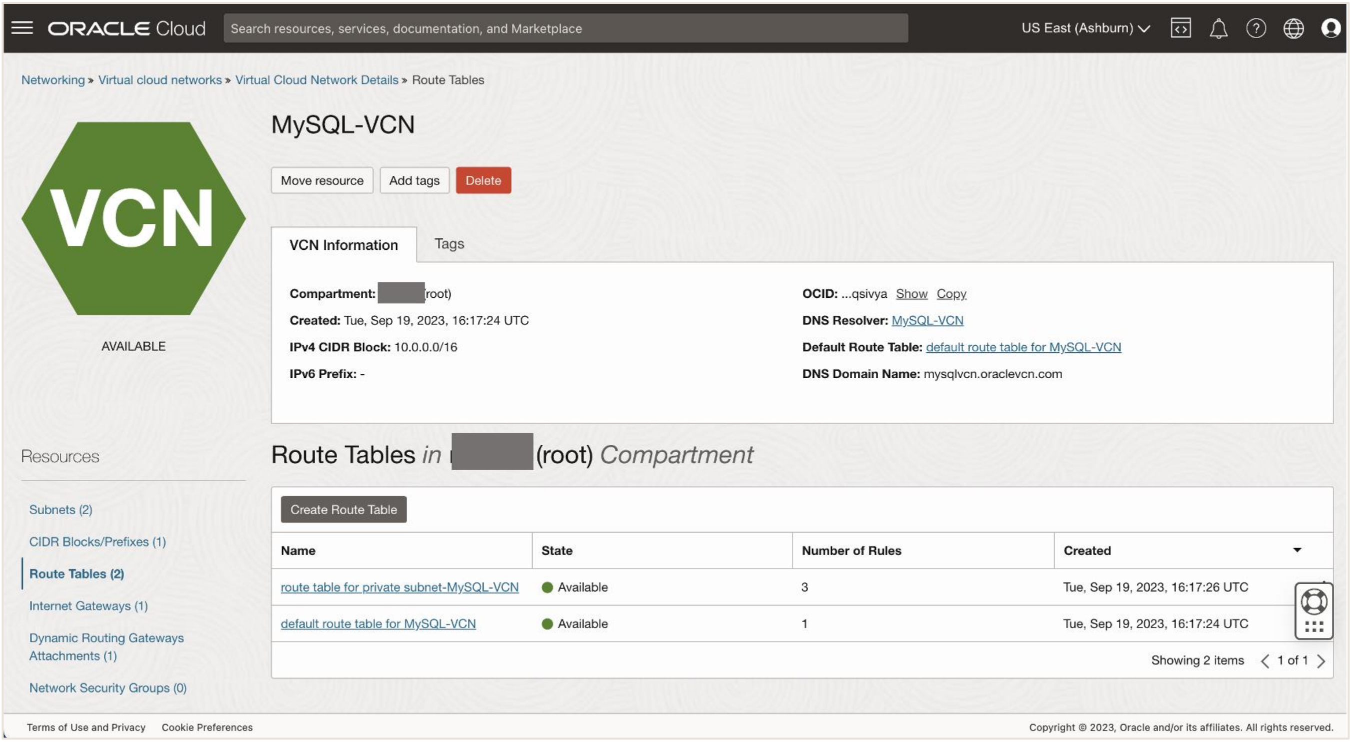
23. On the private subnet route table page, click **Add Route Rules**.



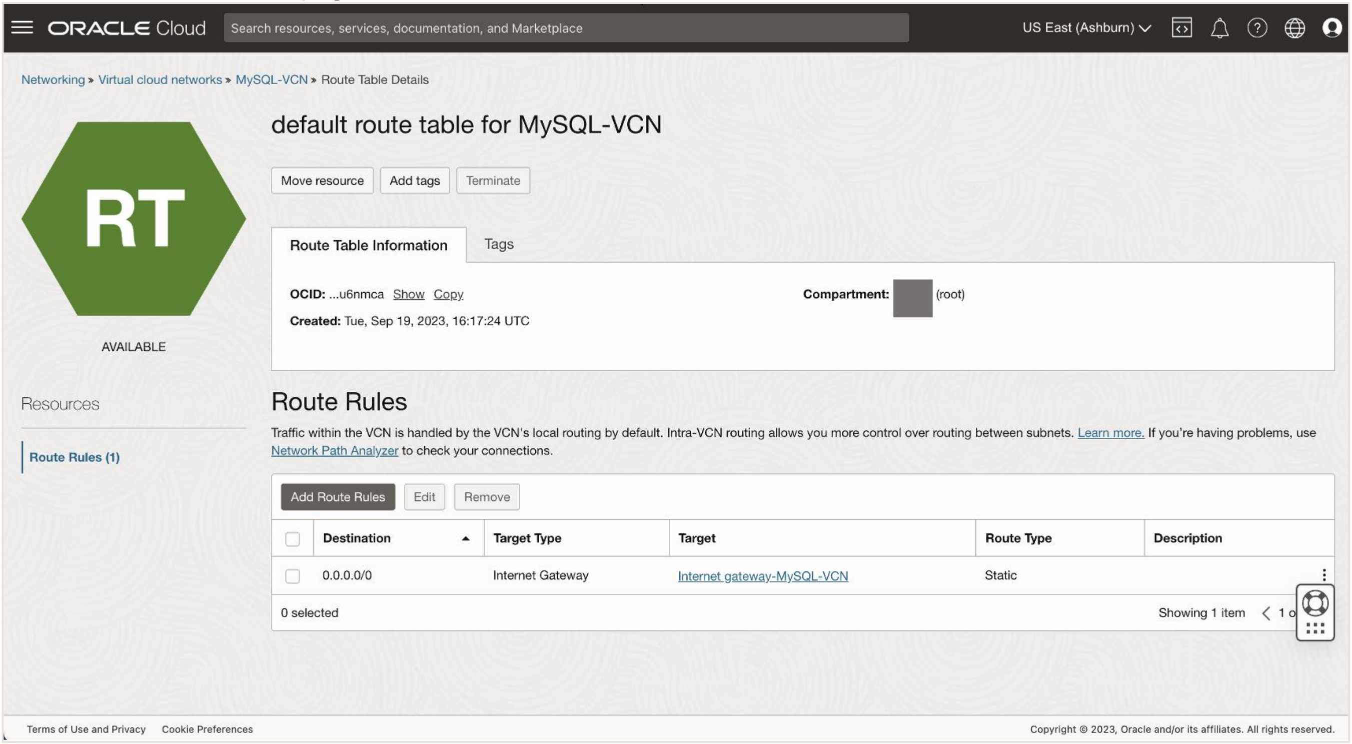
24. Under **Target Type**, select **Dynamic Routing Gateway** from the drop-down list. For **Destination Type**, select **CIDR Block** and for **Destination CIDR Block** - enter your **AWS VPC IPv4 CIDR block** that you will be using to connect to OCI. The AWS VPC CIDR block that will be used for this guide is **10.1.0.0/16**. Click **Add Route Rules** afterwards.



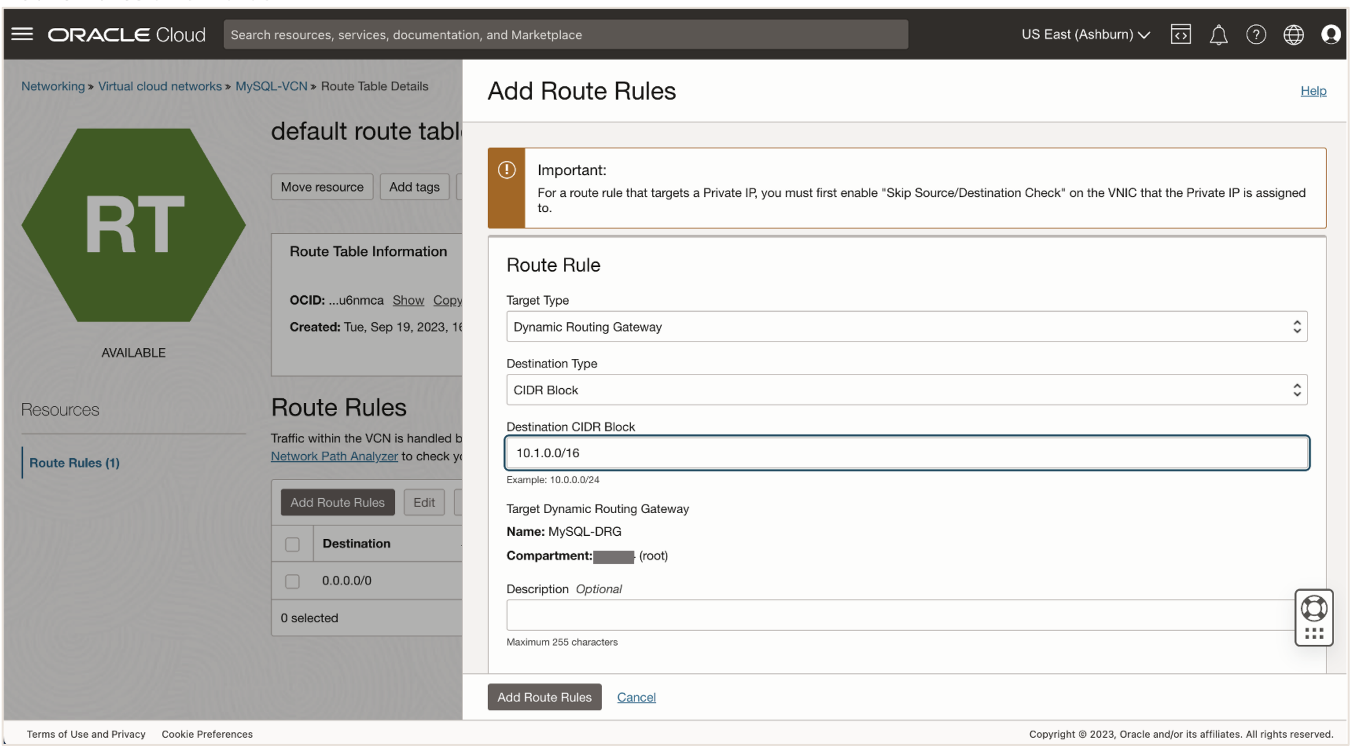
25. Now, repeat the same process for the other route table. Go back to Virtual Cloud Network Details page, click **Route Tables**, and click on **default route table for <vcn-name>**.



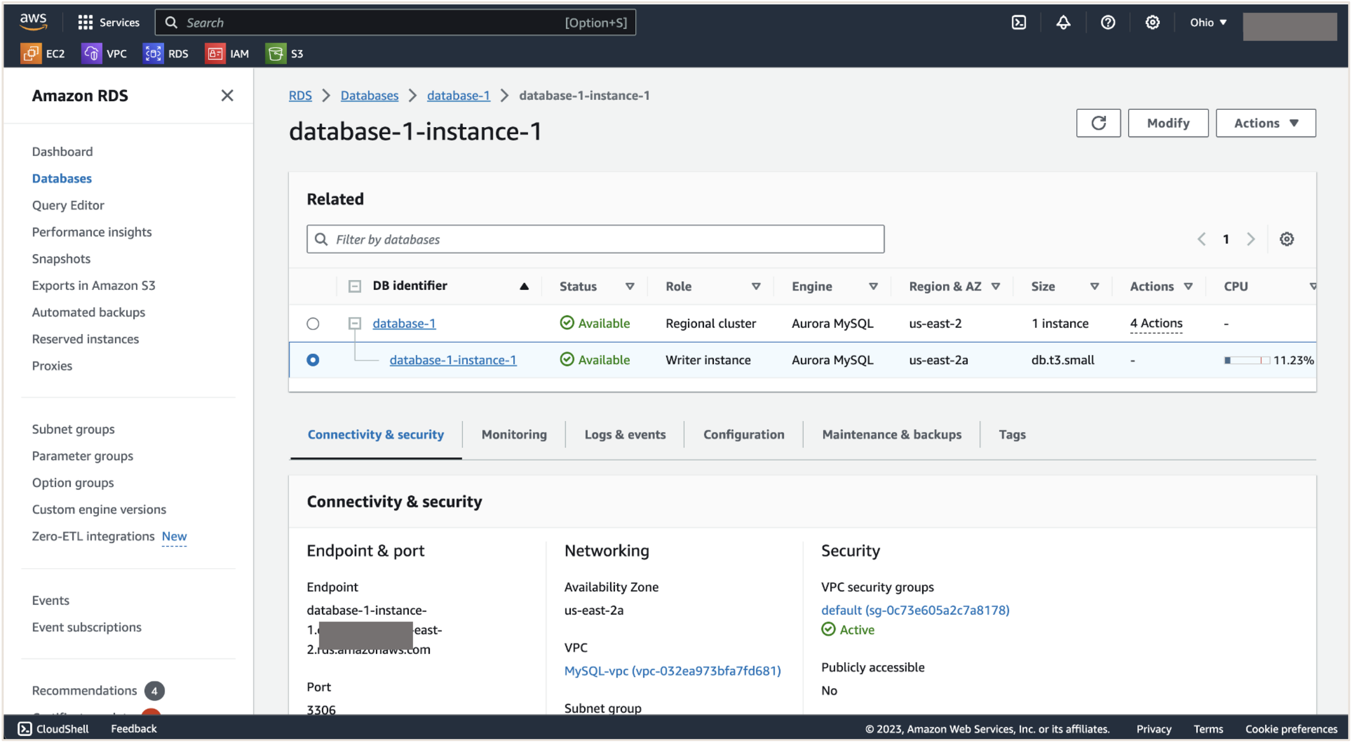
26. On the default route table page, click **Add Route Rules**.



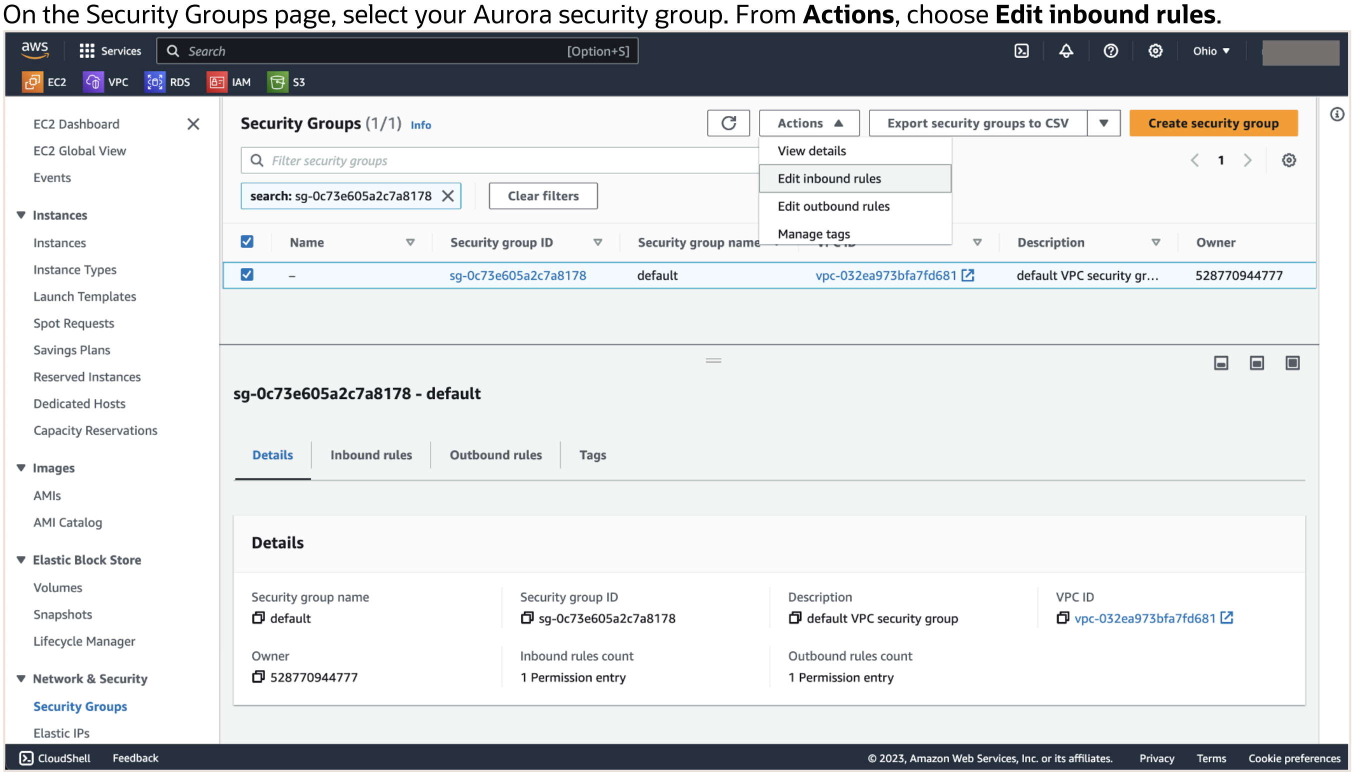
27. Under **Target Type**, **select Dynamic Routing Gateway** from the drop-down list. For **Destination Type**, select **CIDR Block** and for **Destination CIDR Block** - enter your **AWS VPC IPv4 CIDR block** that you will be using to connect to OCI. The AWS VPC CIDR block that will be used for this guide is **10.1.0.0/16**. Click **Add Route Rules** afterwards.



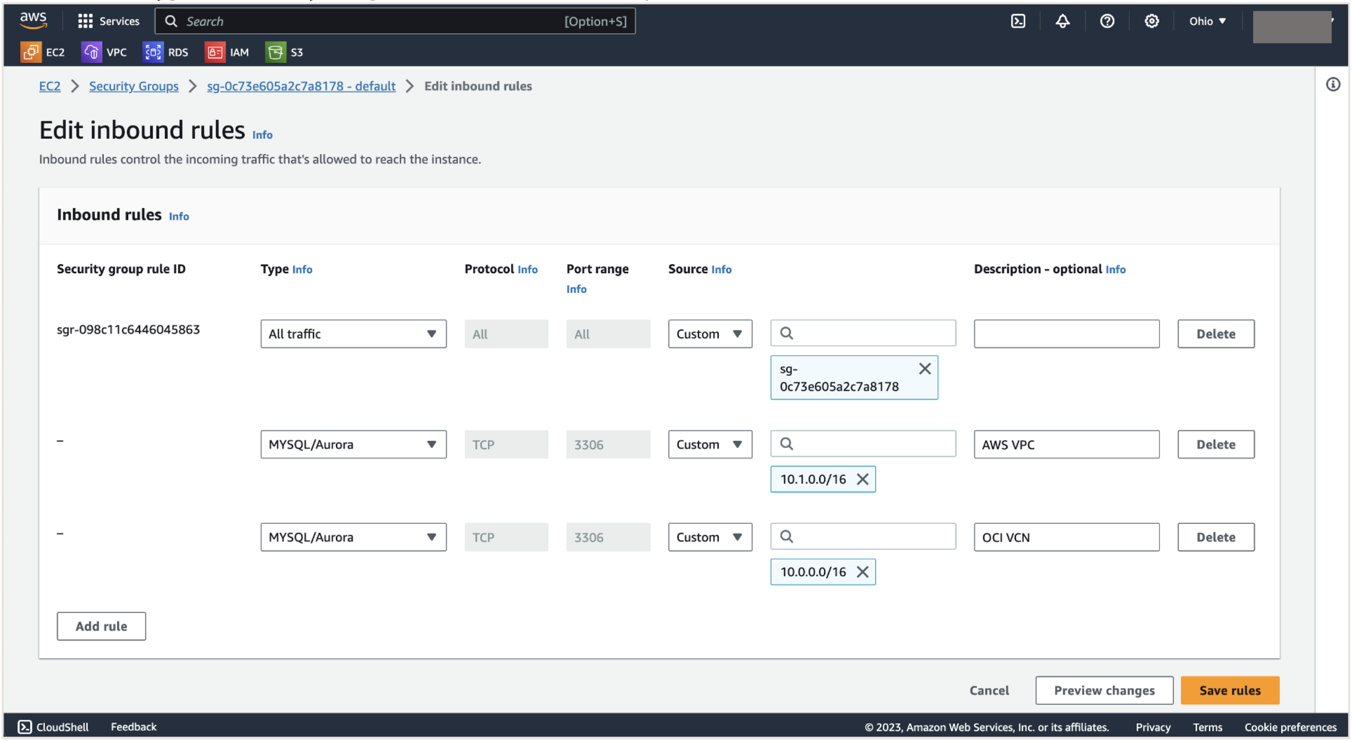
28. Login to AWS to modify the VPC security groups for the Aurora MySQL instance which will allow Aurora to access the HeatWave MySQL instance on OCI and the EC2 instance on AWS. From the main AWS portal, expand the Services menu at the top left of the screen, click **Databases**, click **RDS,** and **select your Aurora instance**. Click **Connectivity & security**, under the **Security** section, look for **VPC security groups** and click on **the security group**. For this guide, our Aurora instance only uses one security group - **default**.



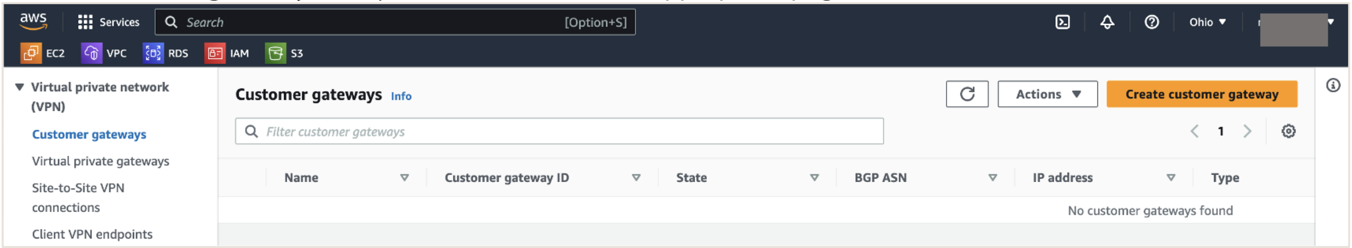
29. On the Security Groups page, select your Aurora security group. From **Actions**, choose **Edit inbound rules**.



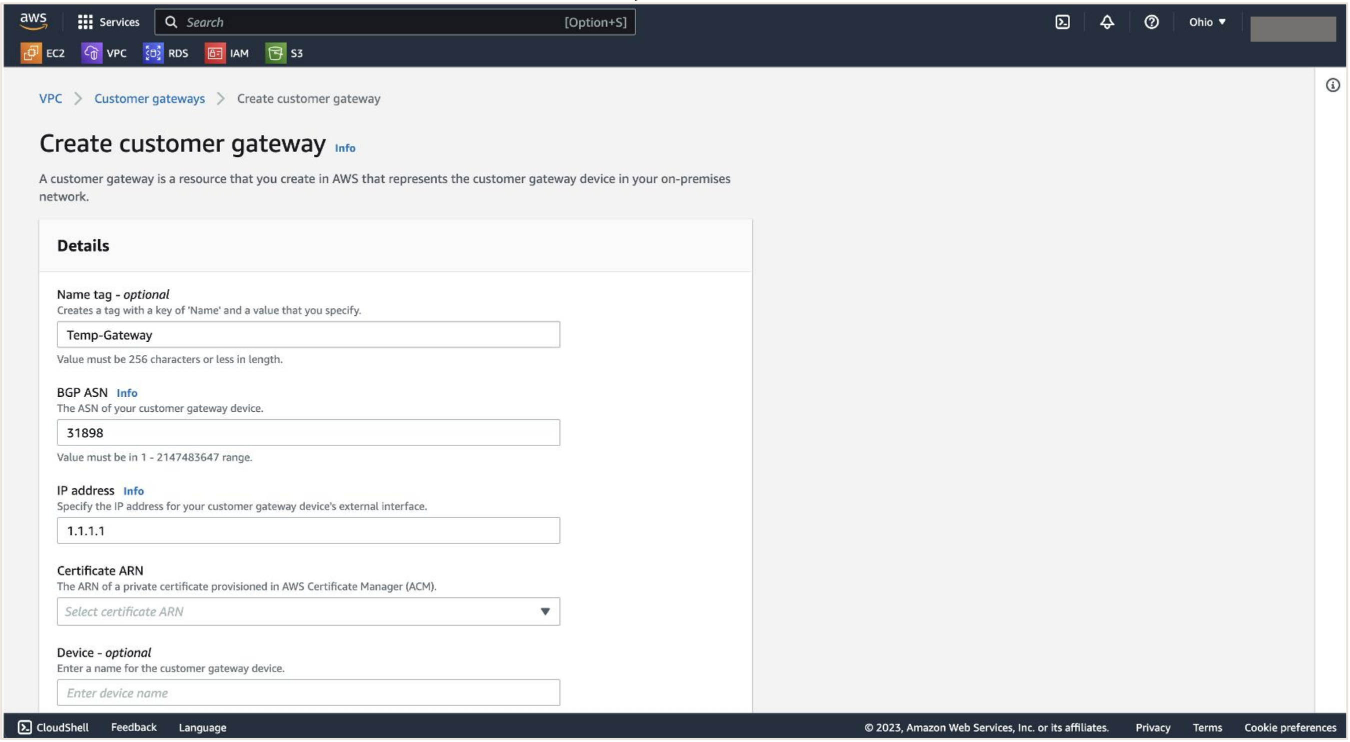
30. Click **Add rule**. Under **Type**, select **MySQL/Aurora**. For **Source**, input the **AWS VPC IPv4 CIDR**. Click **Add rule**. Under **Type**, select **MySQL/Aurora.** For **Source**, input the **OCI VCN IPv4 CIDR block**. Click **Save rules**.



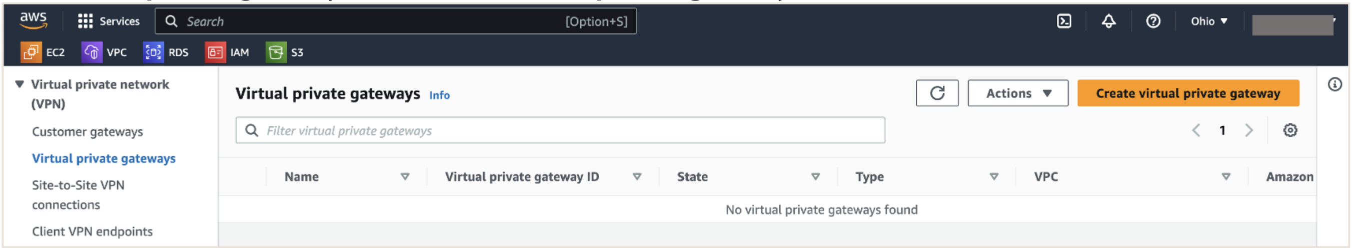
31. From the main AWS Services menu, navigate to **Networking & Content Delivery** and click **VPC**. From the left-hand AWS menu, scroll down and click **Customer Gateways** under Virtual private network (VPN). Click **Create customer gateway** once you have landed on the appropriate page.



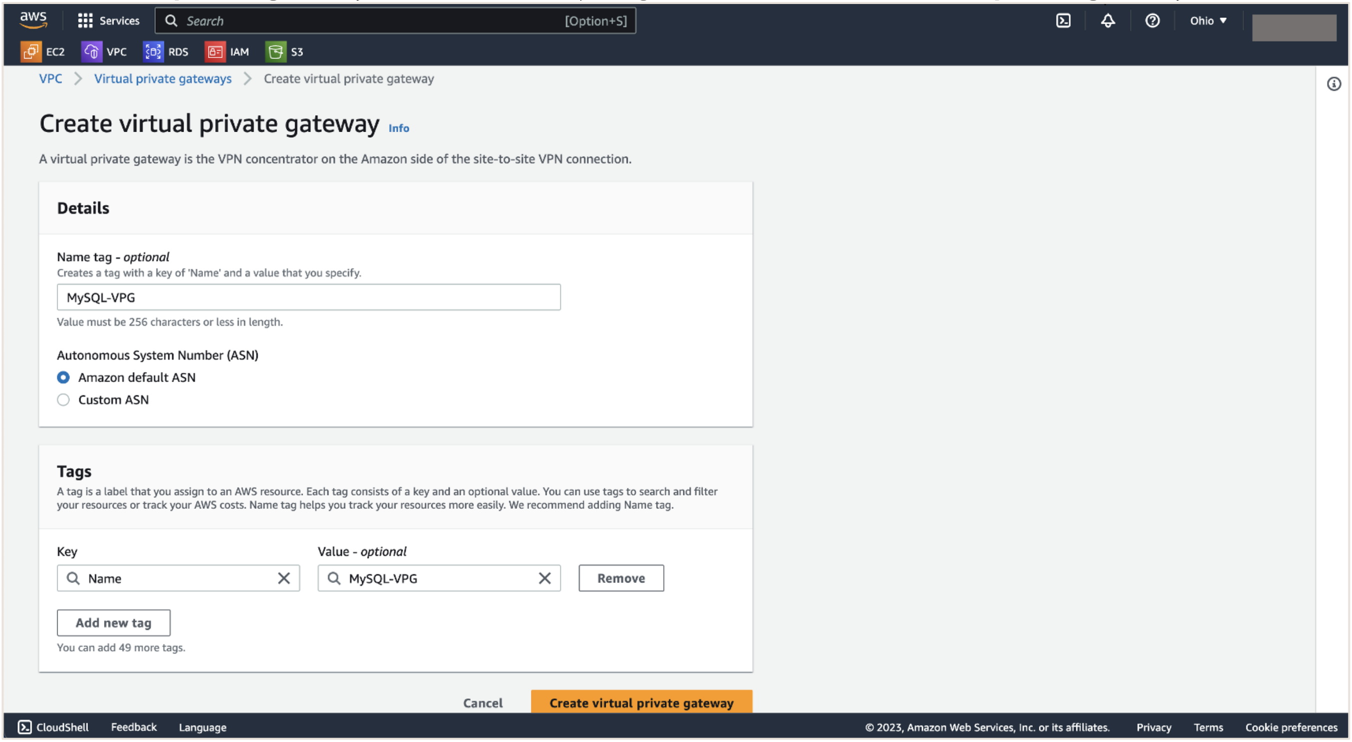
32. Enter a **temporary customer gateway name**. For **BGP ASN** input **31898** and for **IP address** enter **1.1.1.1**. Leave the rest as-is and click **Create Customer Gateway**.



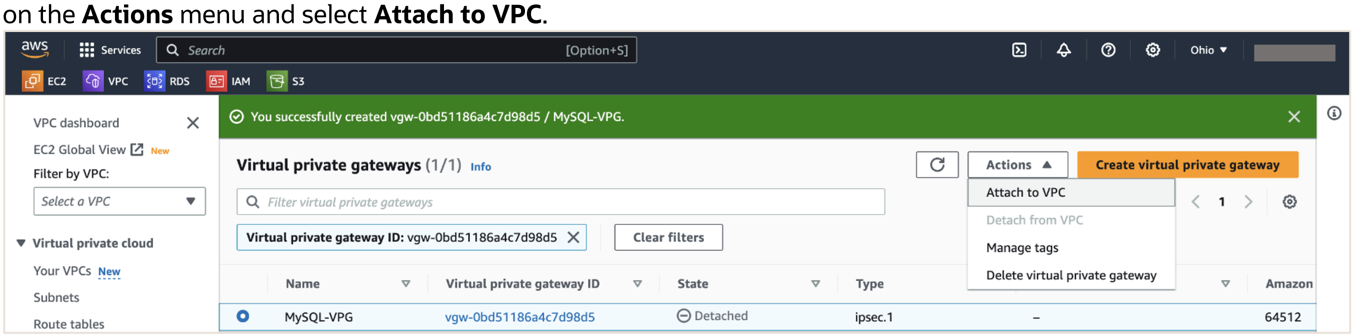
33. From the Customer gateways page, scroll down on the left-hand AWS menu. Under Virtual private network click **Virtual private gateways**. Click **Create virtual private gateway**.



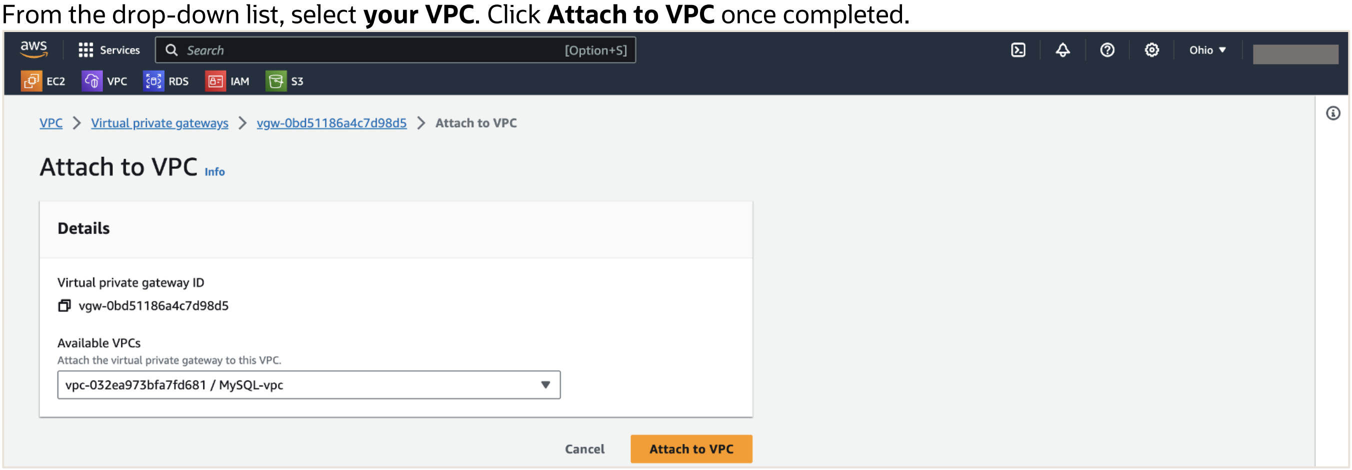
34. Enter a **virtual private gateway name**. Leave everything as-is and click **Create virtual private gateway**.



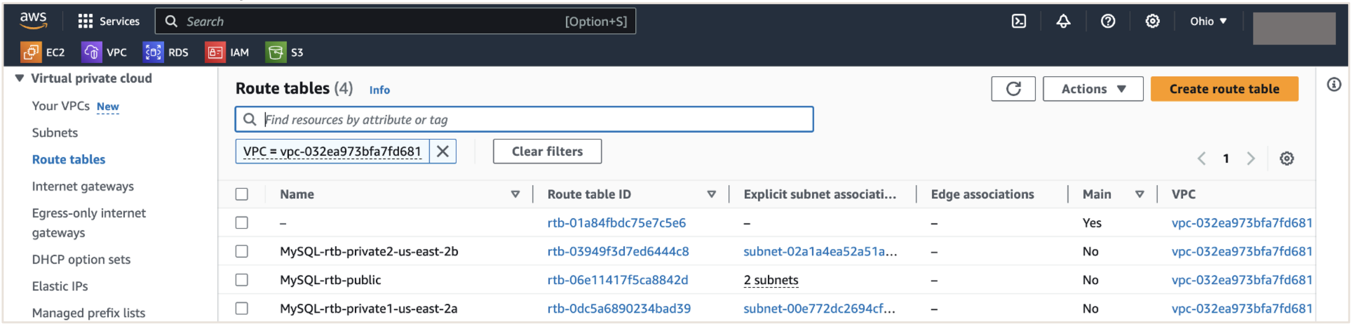
35. While still on the Virtual Private Gateway page, select the **virtual private gateway** that we just created. Click on the **Actions** menu and select **Attach to VPC**.



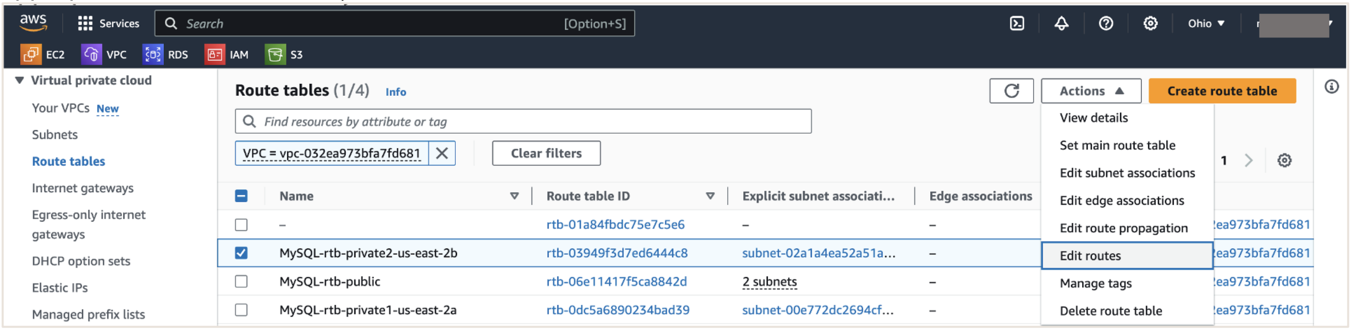
36. From the drop-down list, select **your VPC**. Click **Attach to VPC** once completed.



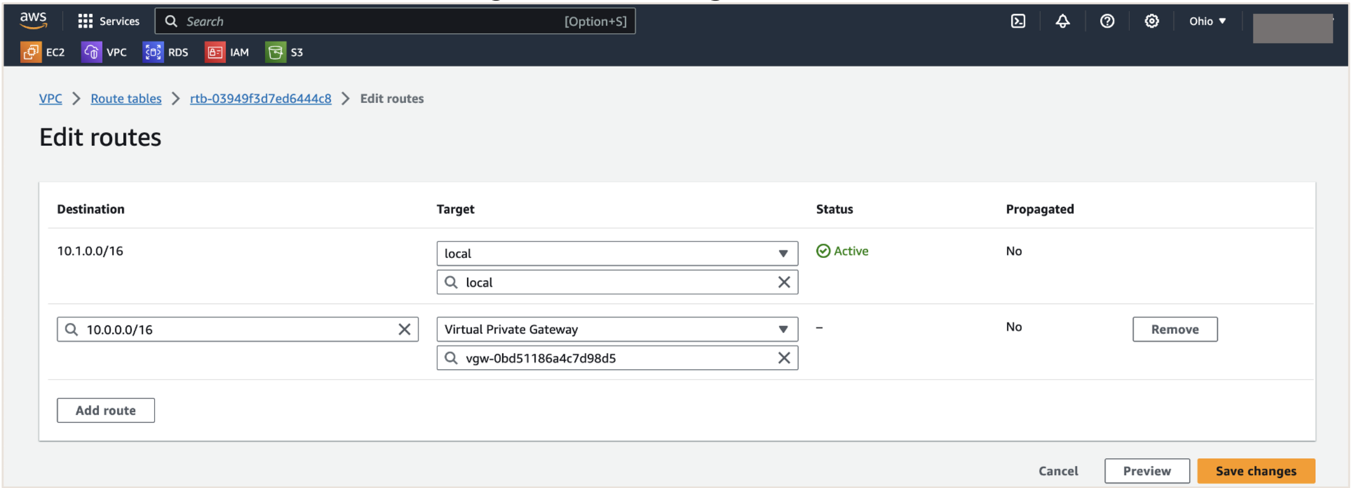
37. Wait until your Virtual private gateway changes its state to **Attached**. It is now time to update the AWS route tables - similar to what we did on OCI. From the Virtual private gateways page, scroll up on the left-hand AWS menu. Under Virtual private cloud, select **Route tables**.



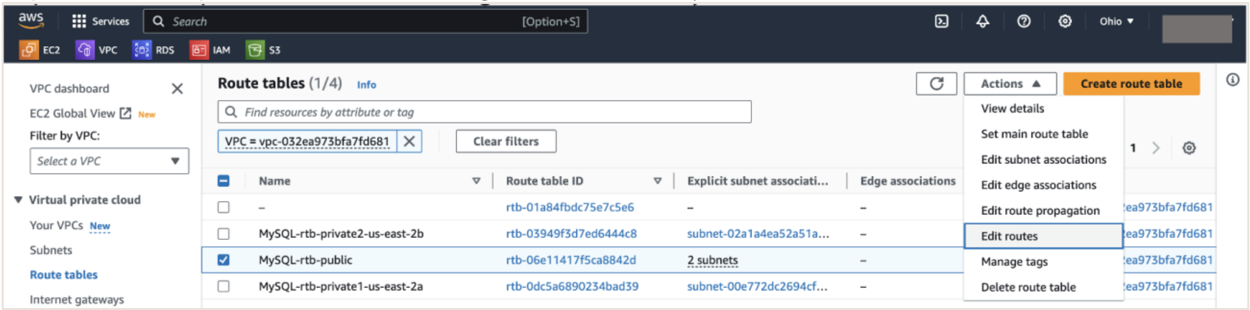
38. For this guide, the main route table (rtb-01a84fbdc75e7c5e6 - the one with no name) is not being used, although we will use the public route table (to deploy on EC2 later) and both private route tables (for Aurora). For each of the route tables that you wish to use, you will need to add an additional route rule. Select the appropriate route table one-by-one and from the **Actions** menu, click **Edit routes**.

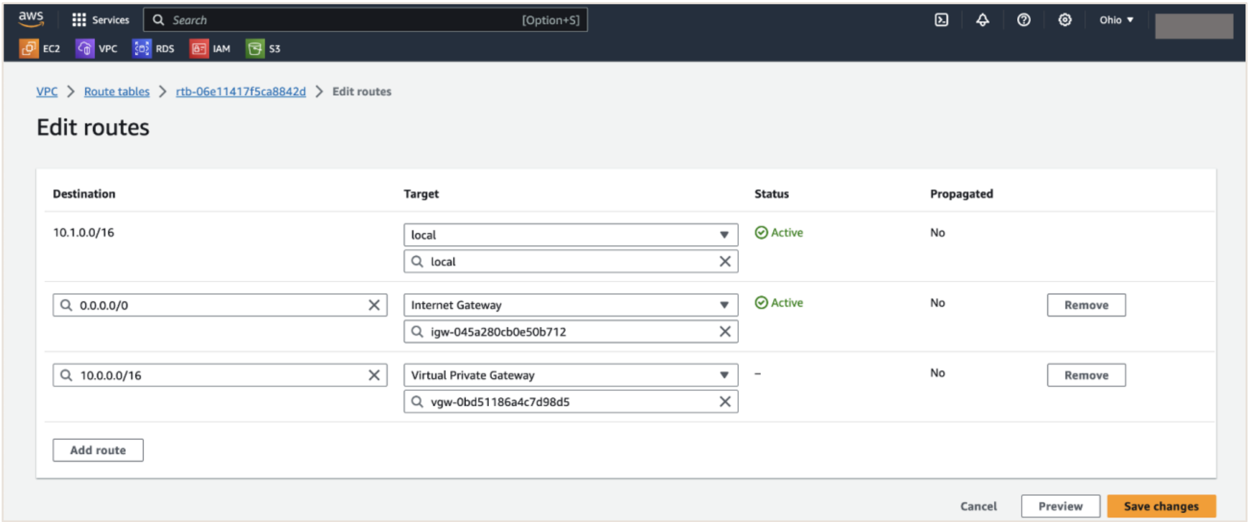


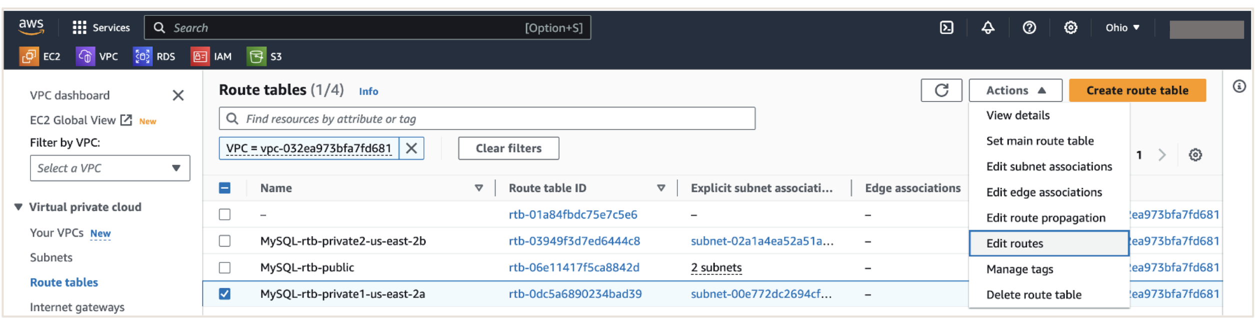
39. Click **Add route** and under the **Destination**, input your **OCI VCN CIDR block** that you are using when you created your OCI VCN (the guide uses OCI VCN CIDR block of **10.0.0.0/16**). Afterwards, for **Target**, click **Virtual Private Gateway** from the drop-down list and **select your Virtual Private Gateway**. Once your route has been added as shown in the below image, click **Save changes**.

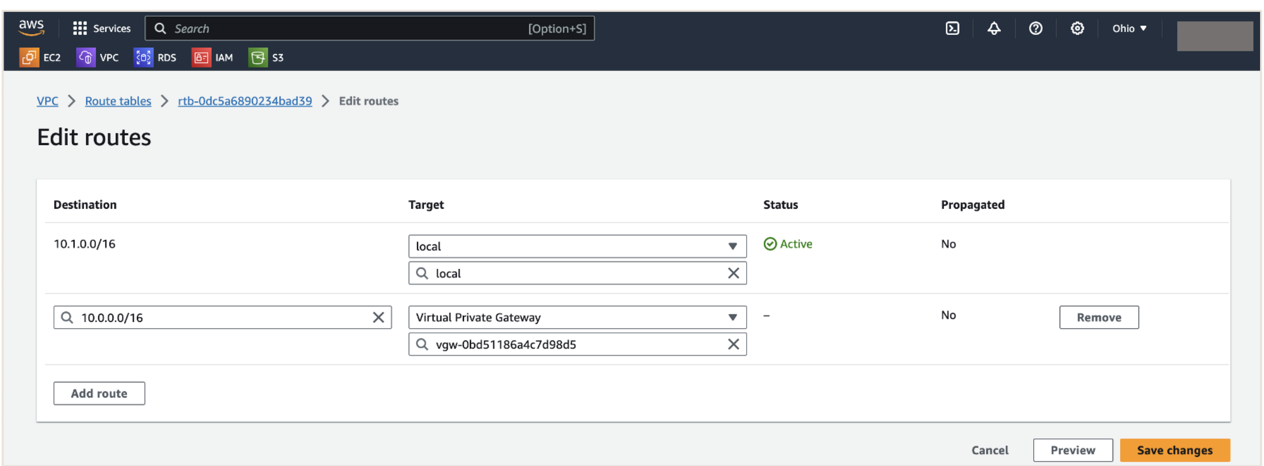


40. Repeat the same process for the remaining route tables that you will use.

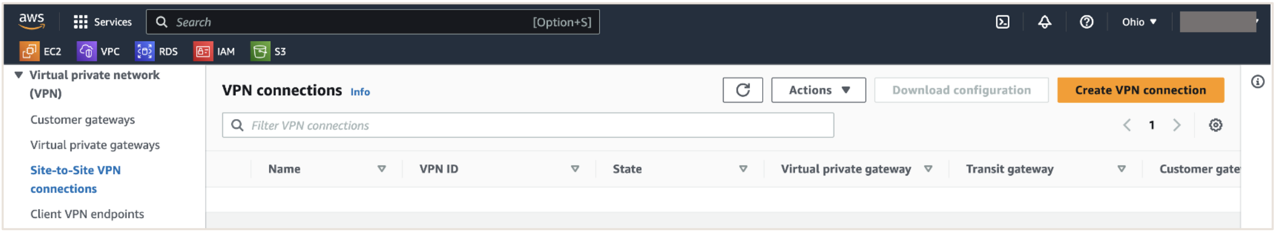




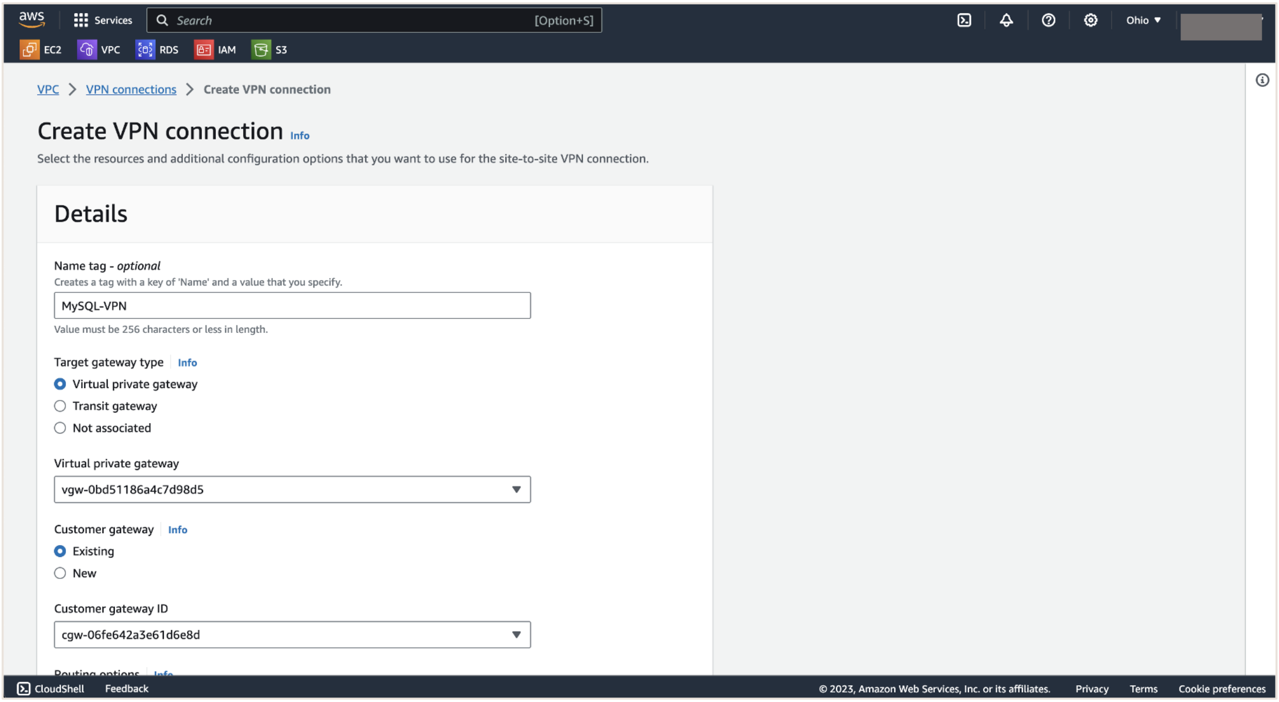


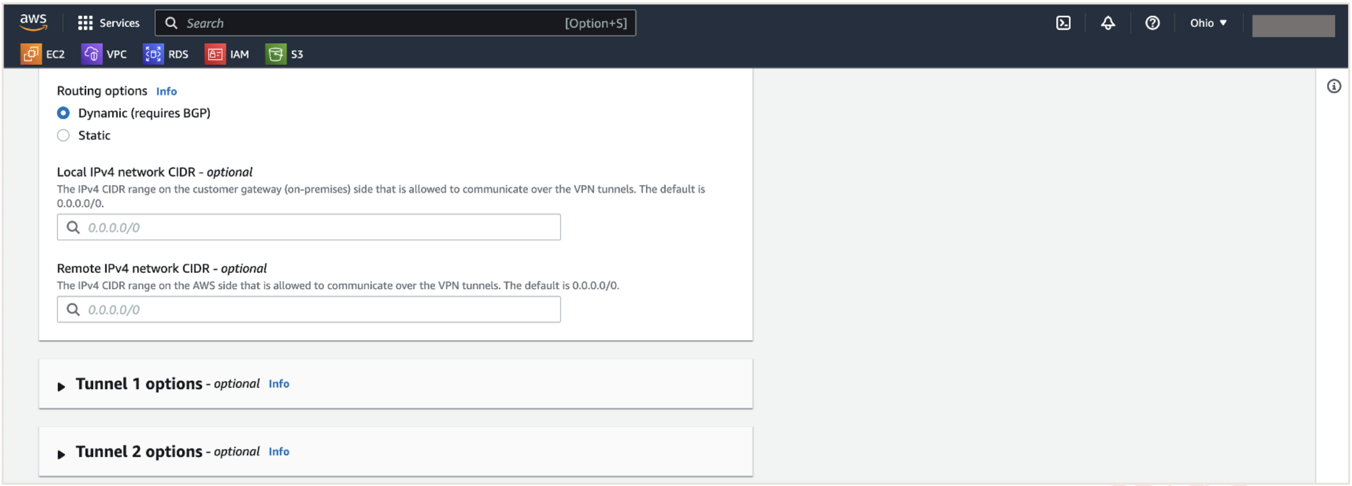


41. After you have updated all your route tables on AWS, from the left-hand menu, scroll down and click **Site-to-Site VPN Connections** under Virtual Private Network (VPN). Once on the appropriate page, click **Create VPN Connection**.



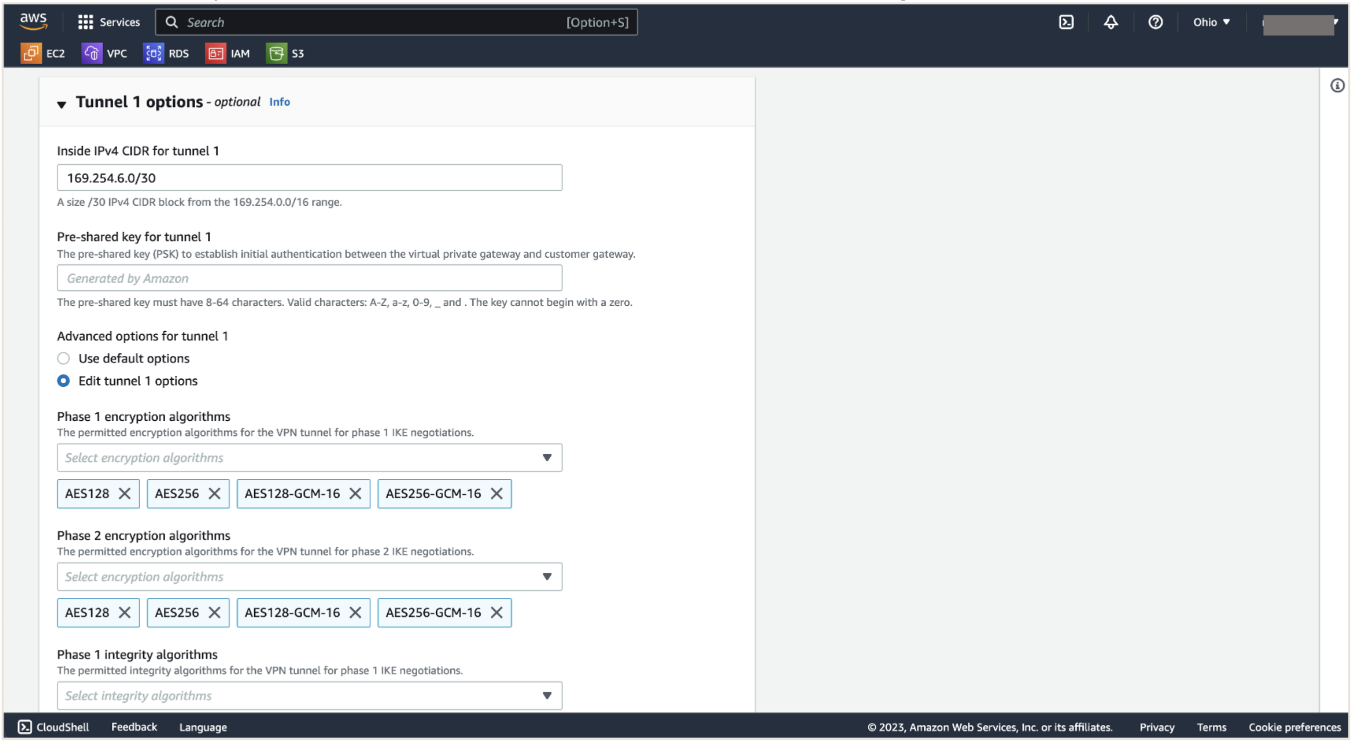
42. Give a **VPN connection name**, for **Target gateway type** select **Virtual private gateway**. Under **Virtual private gateway** drop-down - select **the VPG that we had created earlier**. For **Customer gateway** select **Existing** and under the **Customer gateway ID** drop-down - select **the temporary Customer Gateway that we had created earlier**. Under **Routing options** select **Dynamic (requires BGP)**. Leave the **Local and Remote IPv4 network CIDR** fields blank.



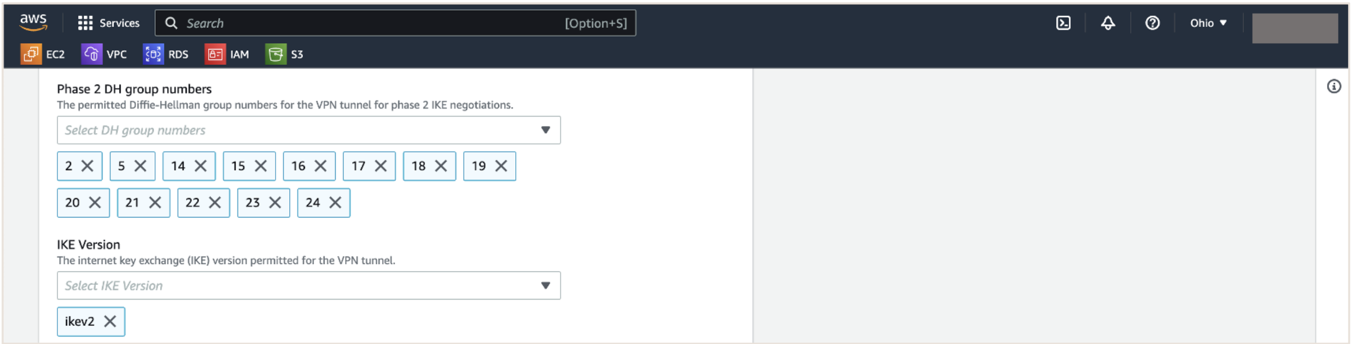


43. While still on the Create VPN Connection page, expand the **Tunnel 1 options**. **Choose a /30 CIDR from within the link local 169.254.0.0/16 range**. Input the full CIDR in the **Inside IPv4 CIDR for Tunnel 1** field. The guide uses the CIDR block of **169.254.6.0/30**. Ensure that OCI supports the chosen /30 address for the inside tunnel IPs. OCI does not allow you to use the following IP ranges for inside tunnel IPs:
    - 169.254.10.0-169.254.19.255
    - 169.254.100.0-169.254.109.255
    - 169.254.192.0-169.254.201.255

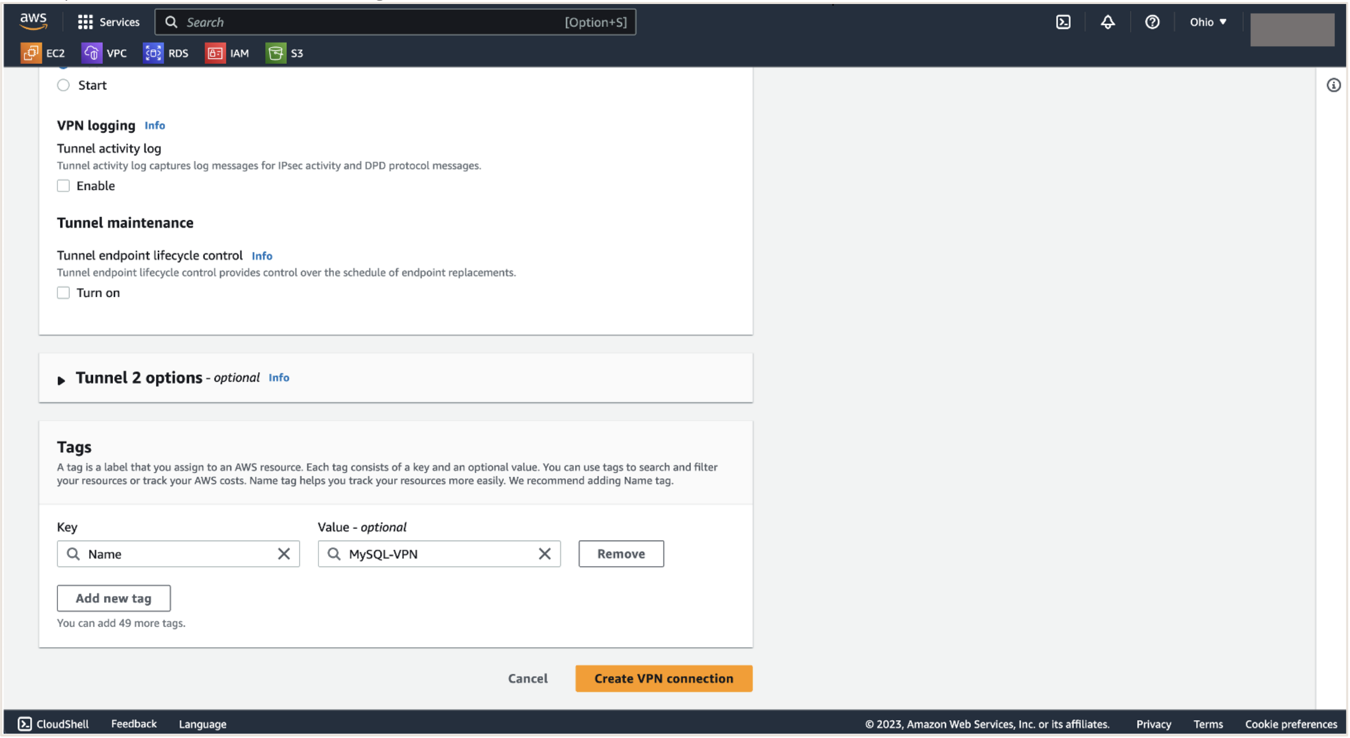
    Under Advanced options for tunnel 1, click the radio button for **Edit tunnel 1 options**.



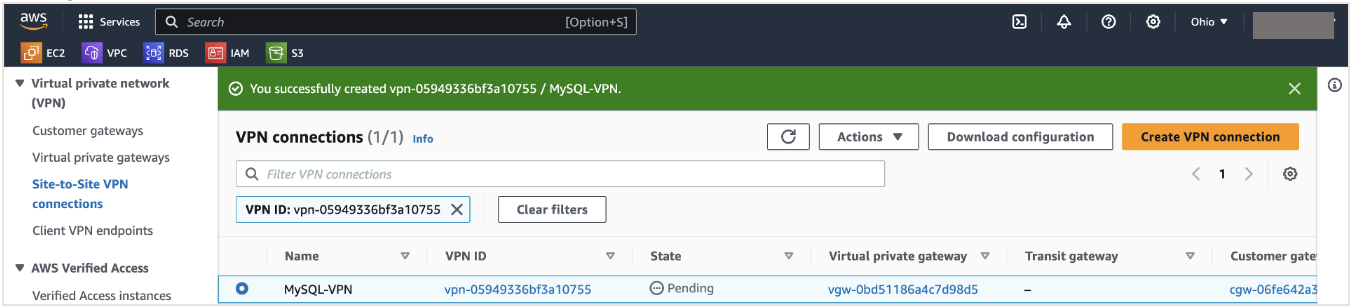
44. Once the tunnel 1 options expand, scroll down and look for **IKE Version**. Click the **X** and remove the **ikev1** field.



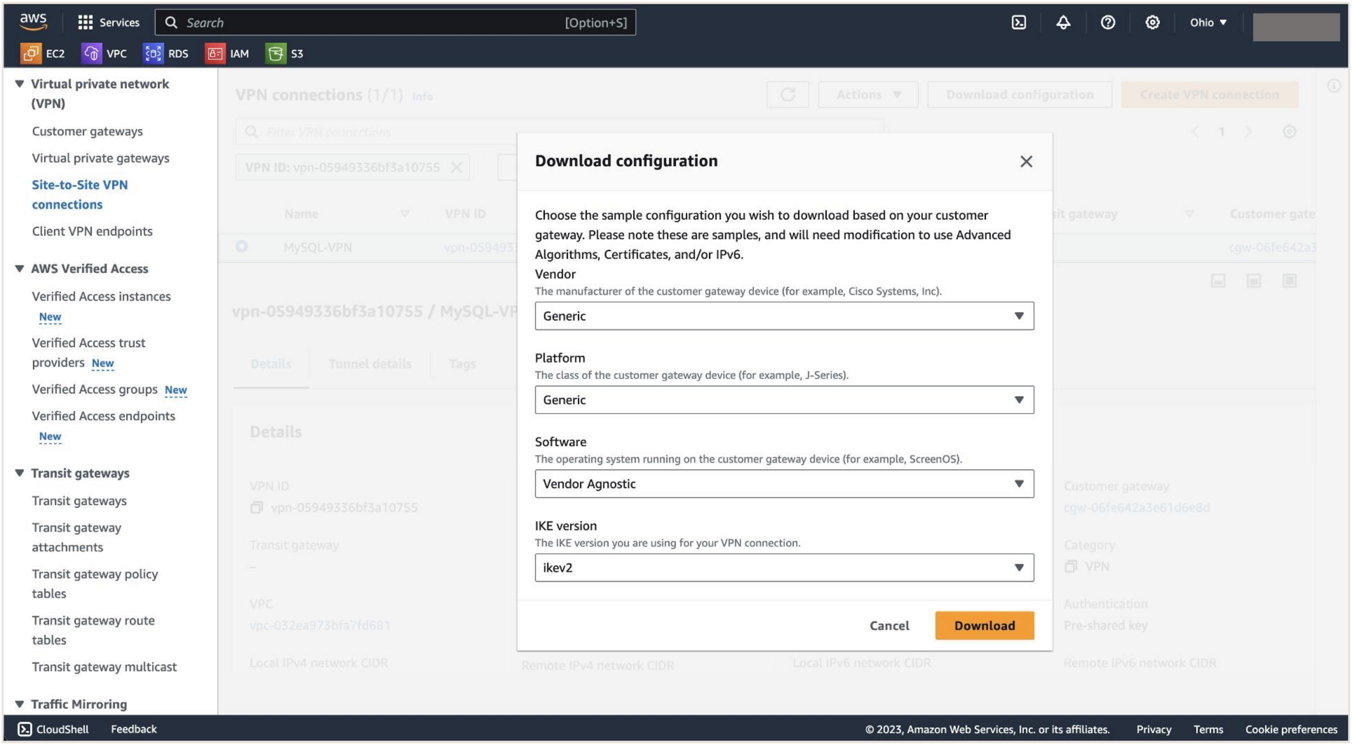
45. After you have finished the configuration, click **Create VPN connection**.



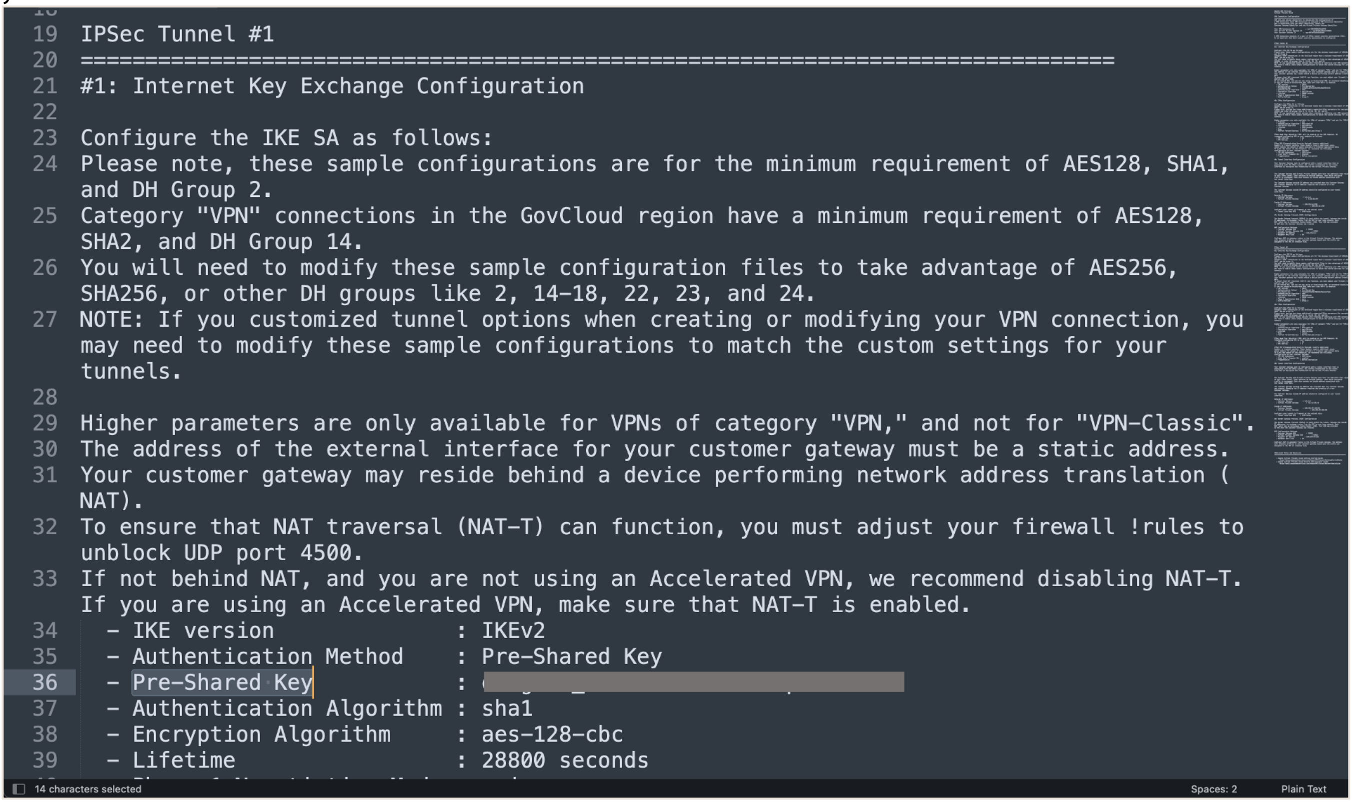
46. On the VPN Connections page, make sure that your VPN connection is selected and click the **Download configuration** button.



47. For **Vendor** and **Platform**, select **Generic**. For **IKE version**, select **ikev2**. Click **Download** afterwards.

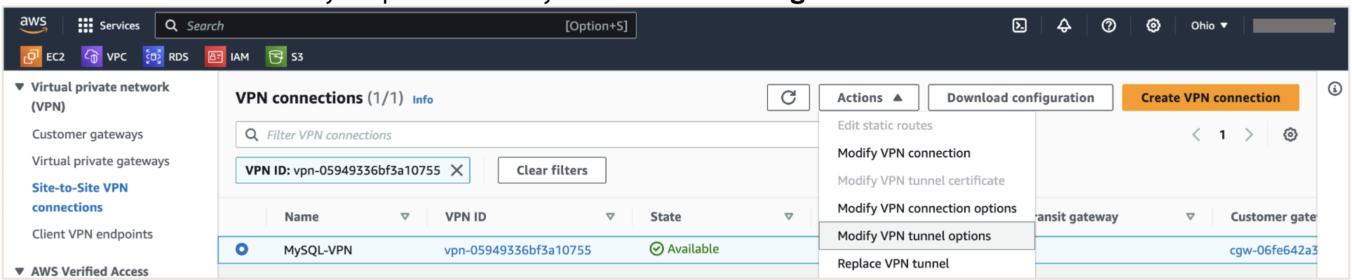


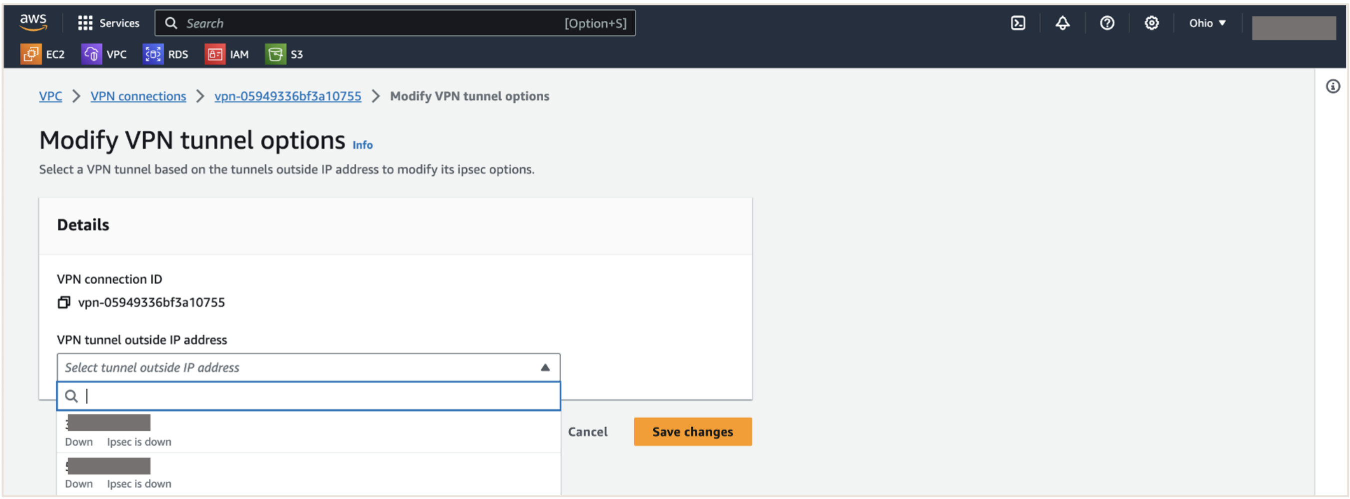
48. Open the downloaded configuration file in your text editor of choice. Look under **IPSec Tunnel #1**, section **#1 Internet Key Exchange Configuration**. Here you find your automatically generated **Pre-Shared Key** for your tunnel. Save this value.

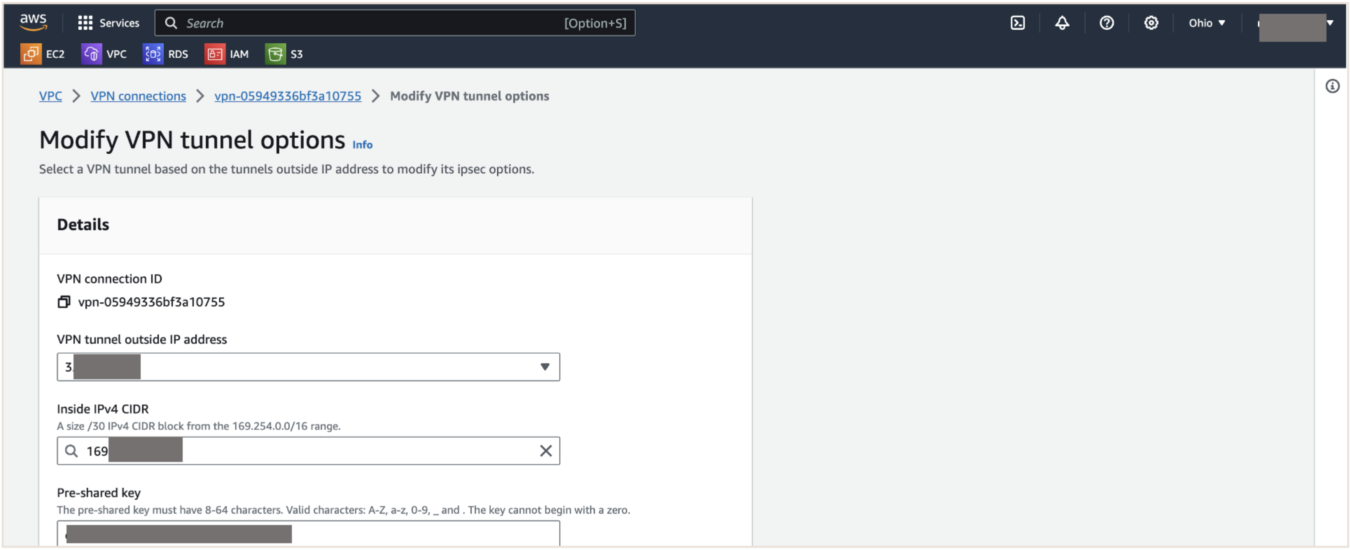


```
IPSec Tunnel #1
================================================================================
#1: Internet Key Exchange Configuration

Configure the IKE SA as follows:
Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and DH Group 2.
Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group 14.
You will need to modify these sample configuration files to take advantage of AES256, SHA256, or other DH groups like 2, 14-18, 22, 23, and 24.
NOTE: If you customized tunnel options when creating or modifying your VPN connection, you may need to modify these sample configurations to match the custom settings for your tunnels.

Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".
The address of the external interface for your customer gateway must be a static address.
Your customer gateway may reside behind a device performing network address translation (NAT).
To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall !rules to unblock UDP port 4500.
If not behind NAT, and you are not using an Accelerated VPN, we recommend disabling NAT-T. If you are using an Accelerated VPN, make sure that NAT-T is enabled.
  - IKE version              : IKEv2
  - Authentication Method    : Pre-Shared Key
  - Pre-Shared Key           : [redacted]
  - Authentication Algorithm : sha1
  - Encryption Algorithm     : aes-128-cbc
  - Lifetime                 : 28800 seconds
```

Note: AWS might generate a pre-shared key using the period or underscore characters (. or _). OCI does not support using those characters in a pre-shared key. A key that includes these values must be changed. To change your pre-shared key in AWS for a tunnel, select your VPN connection, click the **Actions** button, then **Modify VPN Tunnel Options**. Select the **IPSec Tunnel #1 Virtual Private Gateway outside IP address** from the drop-down (you can find this in the AWS downloaded configuration file). Remove the period or underscore characters from your pre-shared key and click **Save changes**.







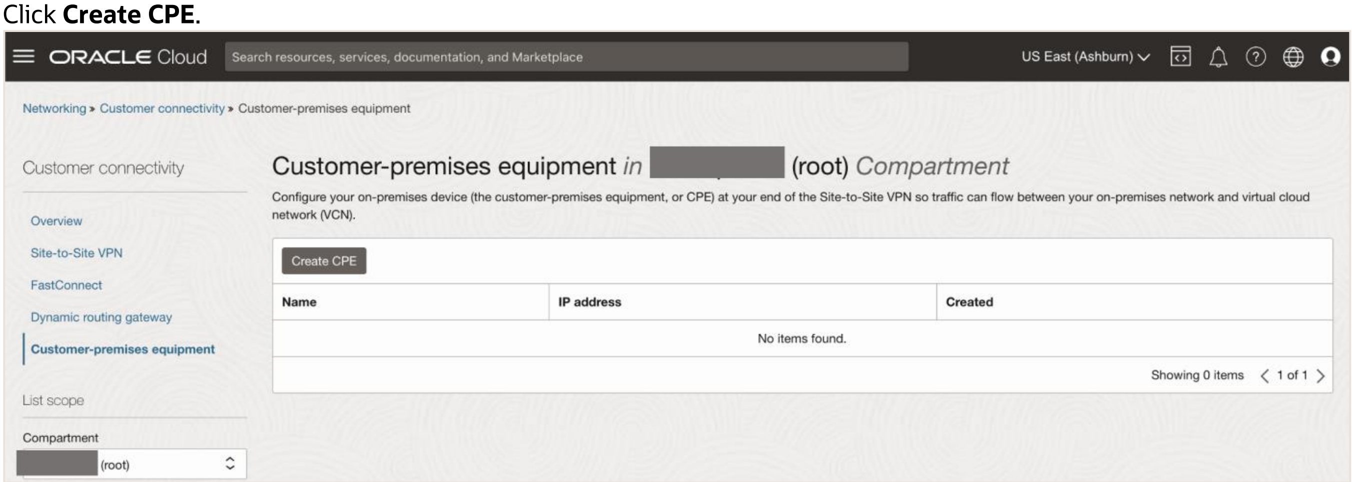
49. While still under Tunnel 1 in the downloaded configuration, scroll down to section **#3 Tunnel Interface Configuration**. Here, note down all the values for **Outside IP Addresses** and **Inside IP Addresses**.

    Scroll down to section **#4: Border Gateway Protocol (BGP) Configuration** and note down the **Virtual Private Gateway ASN** value.

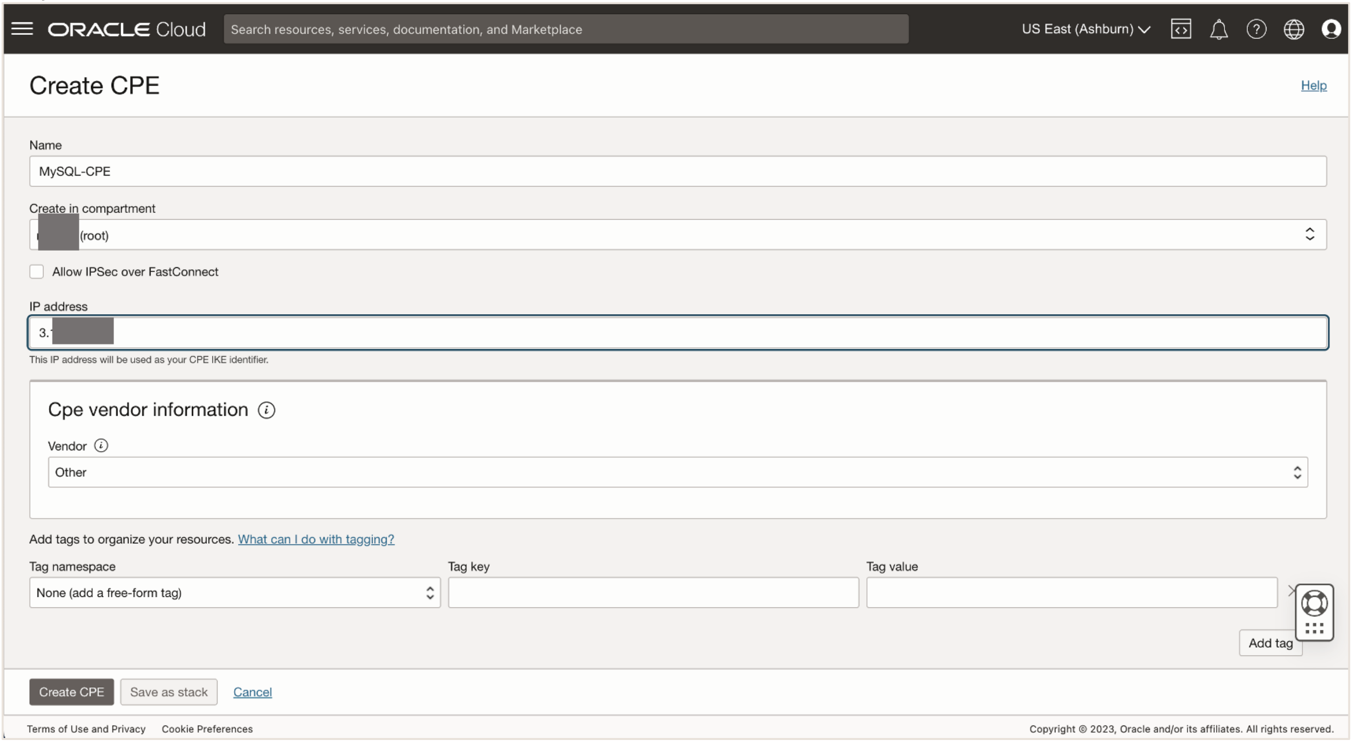
```
The Customer Gateway outside IP address was provided when the Customer Gateway
was created. Changing the IP address requires the creation of a new
Customer Gateway.

The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
  - Customer Gateway            : 1.1.1.1
  - Virtual Private Gateway        : 3.[redacted]

Inside IP Addresses
  - Customer Gateway            : 169.[redacted]
  - Virtual Private Gateway            : 169.[redacted]

Configure your tunnel to fragment at the optimal size:
  - Tunnel interface MTU     : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:

The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside
IP addresses, to exchange routes from the VPC to your home network. Each
BGP router has an Autonomous System Number (ASN). Your ASN was provided
to AWS when the Customer Gateway was created.

BGP Configuration Options:
  - Customer Gateway ASN            : 31898
  - Virtual Private  Gateway ASN            : 64512
  - Neighbor IP Address           : 16[redacted].
  - Neighbor Hold Time       : 30
```

50. Log back in to OCI. From the OCI Navigation menu, navigate to **Networking**, click **Customer connectivity**, and click on **Customer-premises equipment**.

51. Click **Create CPE**.

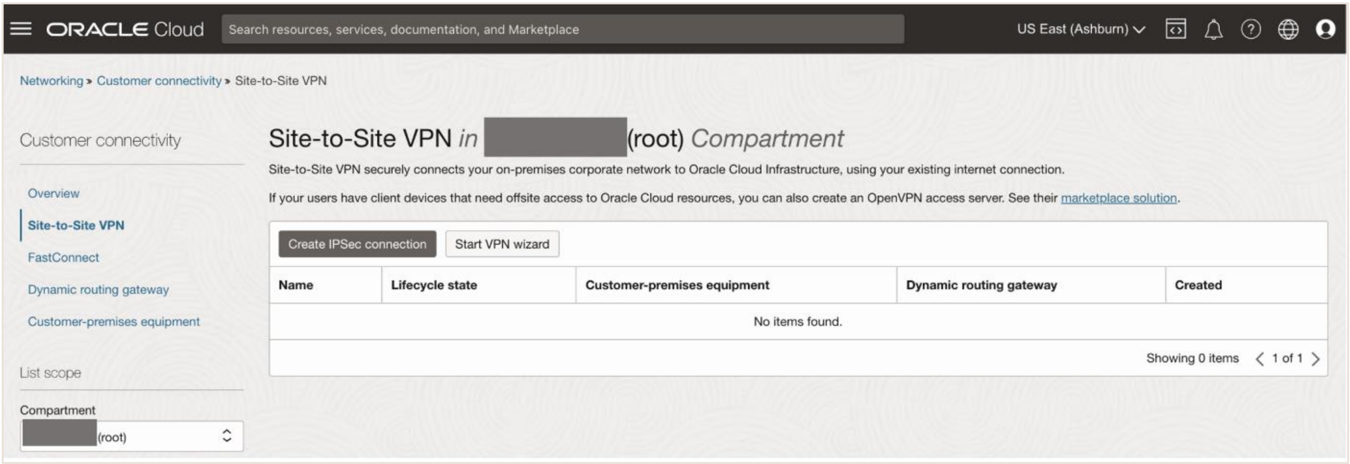


52. Enter a **CPE name**. For the **Public IP address**, input the **Outside IP Address of the Virtual Private Gateway** - you can find this in the configuration file downloaded from AWS. For **CPE Vendor**, select **Other** from the dropdown. Click **Create CPE**.



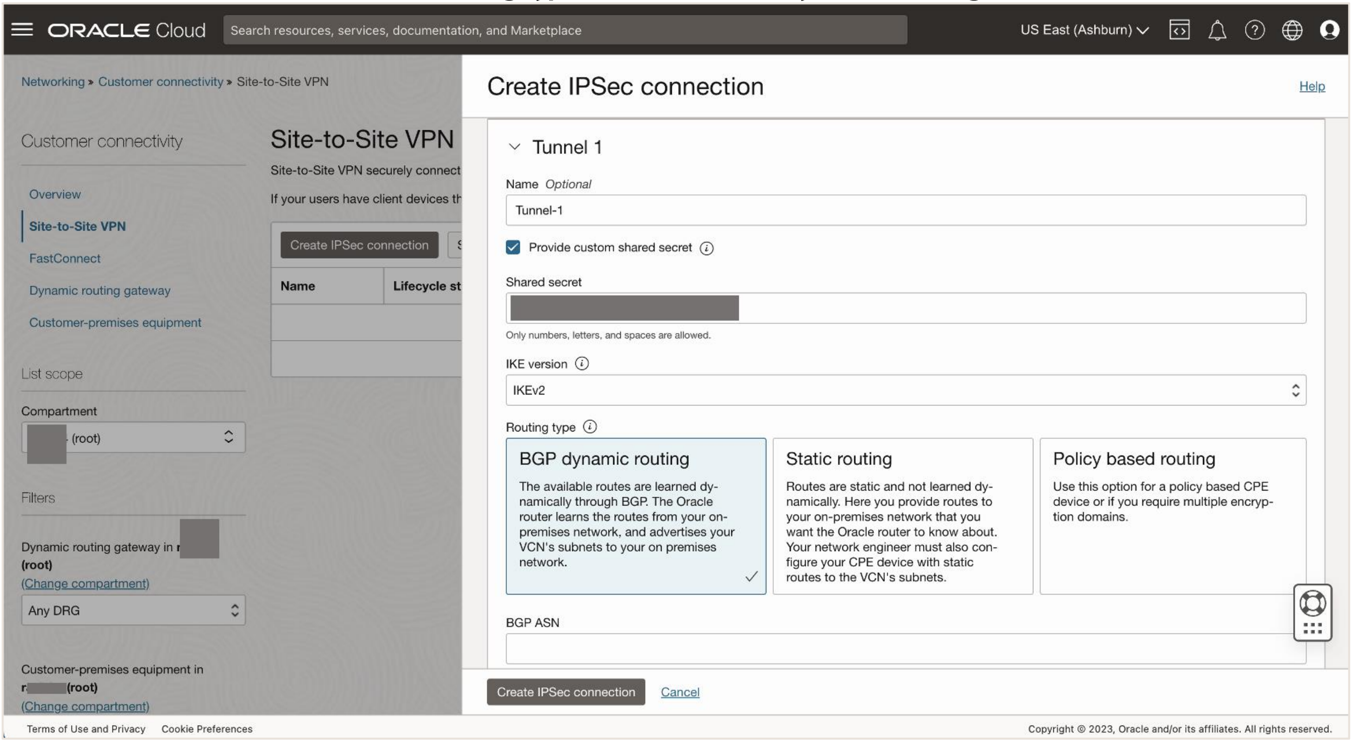
53. From the OCI Navigation menu, navigate to **Networking** and click on **Site-to-Site VPN**.

54. Click **Create IPSec connection**.

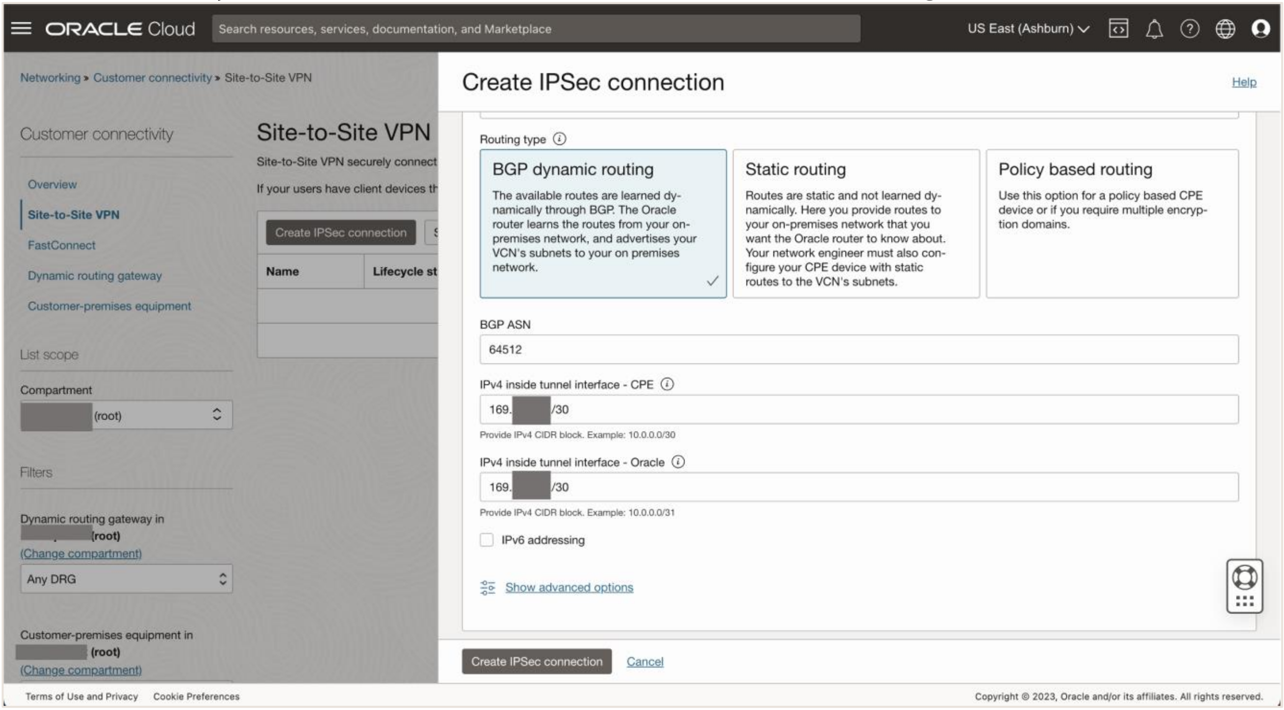


55. Enter a **IPSec connection name**. Under **Customer-premises equipment** dropdown, select the CPE we previously created. For **Dynamic routing gateway compartment** select the DRG we created. For **Routes to your on-premises network**, enter **0.0.0.0/0**.

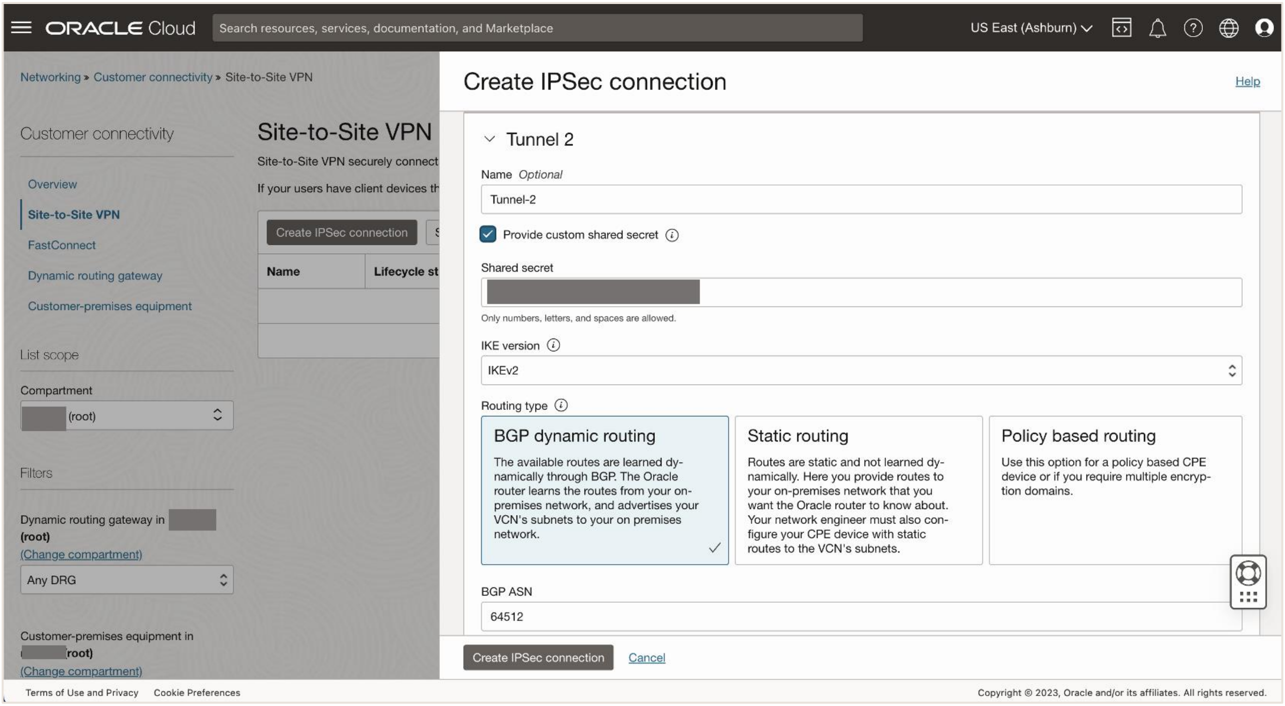
56. While on the Create IPSec connection page, configure your **Tunnel 1**. Enter a **tunnel name**, check the **Provide custom shared secret** box, and input the **Pre-Shared Key** from the AWS VPN configuration file. For **IKE version**, select **IKEv2** and under **Routing type** - make sure **BGP dynamic routing** is selected.



57. Under **BGP ASN**, input the **BGP Virtual Private Gateway ASN** from the AWS VPN configuration file. The default AWS BGP ASN is **64512**. For **IPv4 inside tunnel interface - CPE**, enter the **Inside IP Address of the Virtual Private Gateway**. For **IPv4 inside tunnel interface - Oracle**, enter the **Inside IP Address of the Customer Gateway**. You can find all of this information from the AWS VPN configuration file.

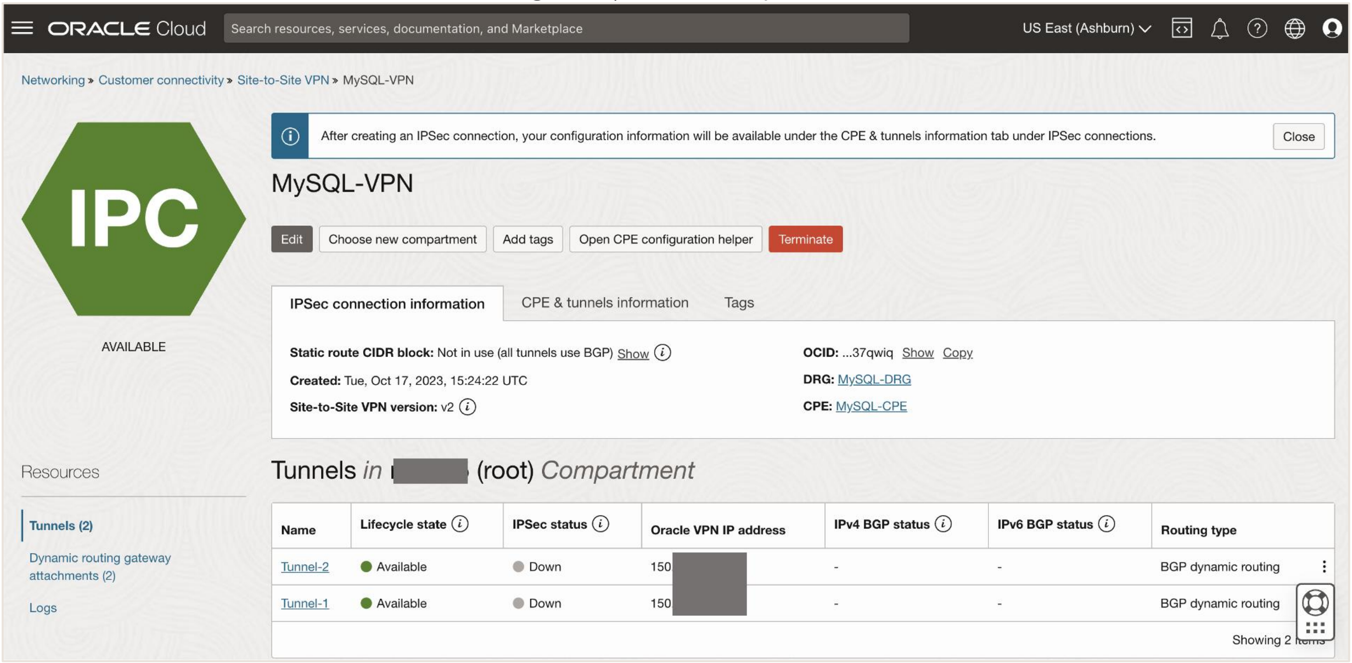


58. Configure your **Tunnel 2** by copying and pasting the same values from Tunnel 1 into Tunnel 2. Click **Create IPSec connection**.

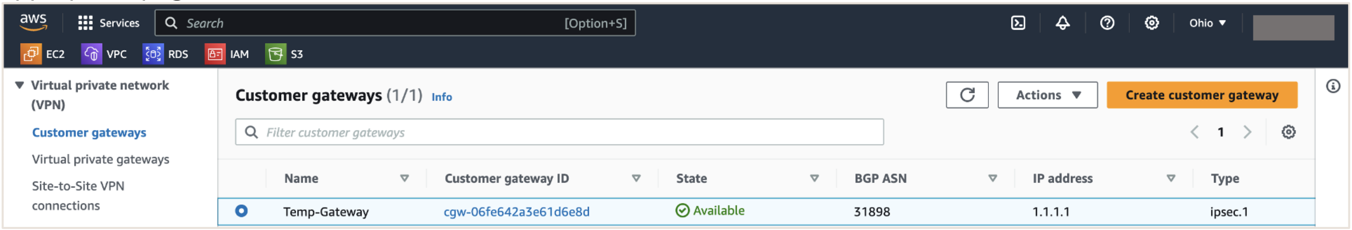


Note: only Tunnel 1 will be used for this VPN connection and migration. We need to configure Tunnel 2 otherwise we cannot click Create IPSec connection.

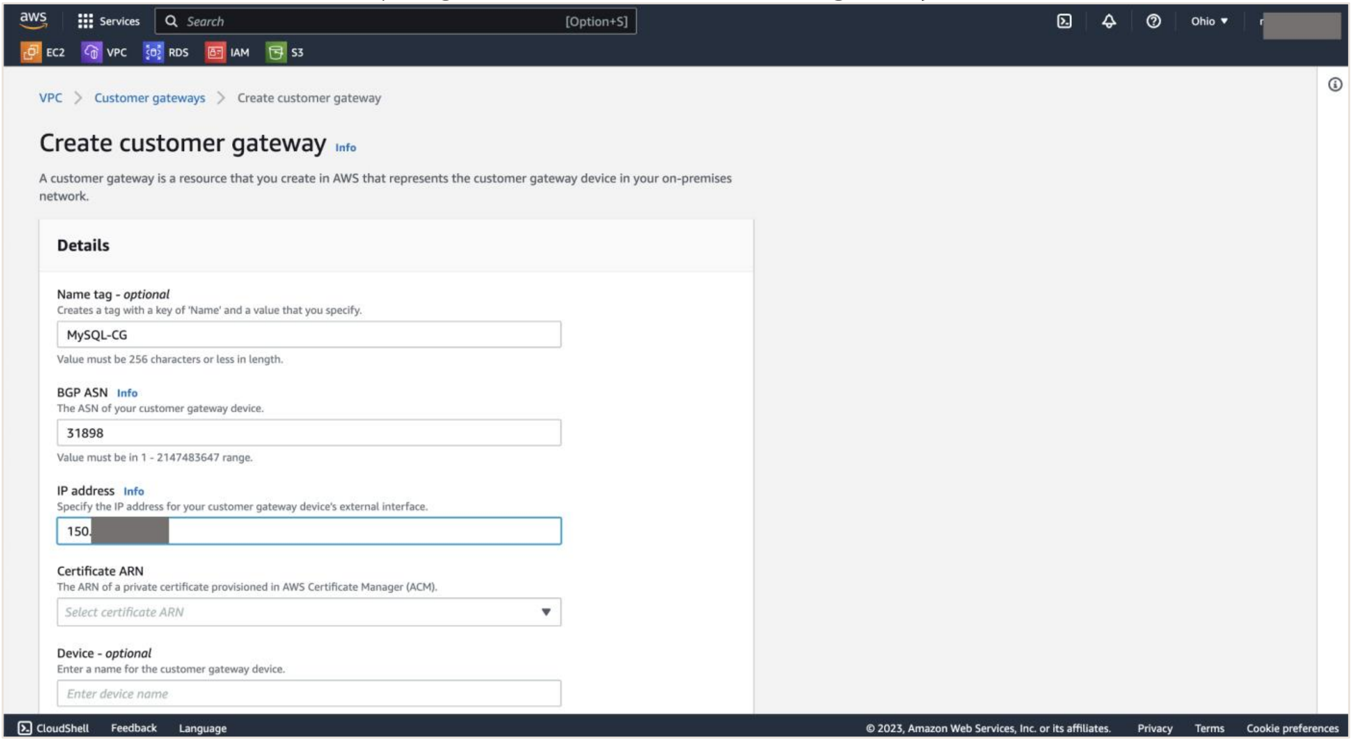
59. After your IPSec connection is provisioned, make note of the **Oracle VPN IP Address** of **Tunnel-1**. This address will be used to create a new customer gateway in the AWS portal.



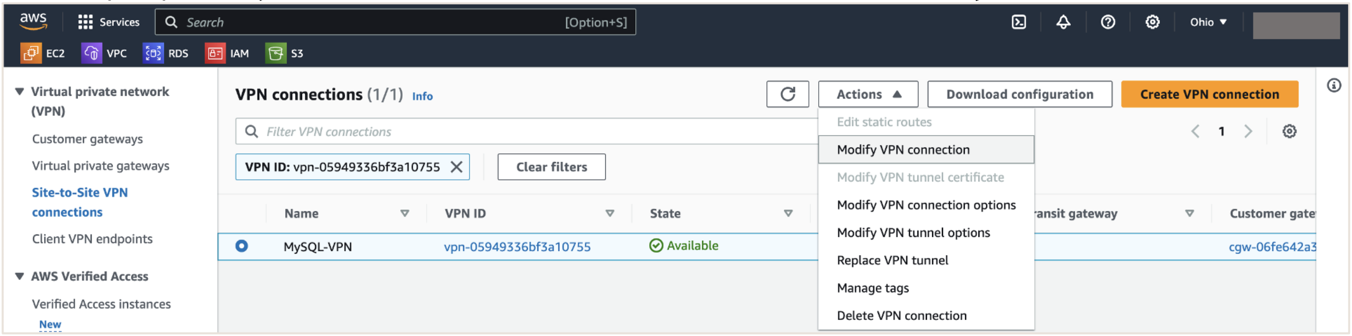
60. Log back in to AWS. Expand the Services menu at the top left of the screen. Navigate to **Networking & Content Delivery** and select **VPC**. From the left-hand menu, scroll down and click **Customer Gateways** under Virtual private network (VPN). Click **Create customer gateway** once you have landed on the appropriate page.



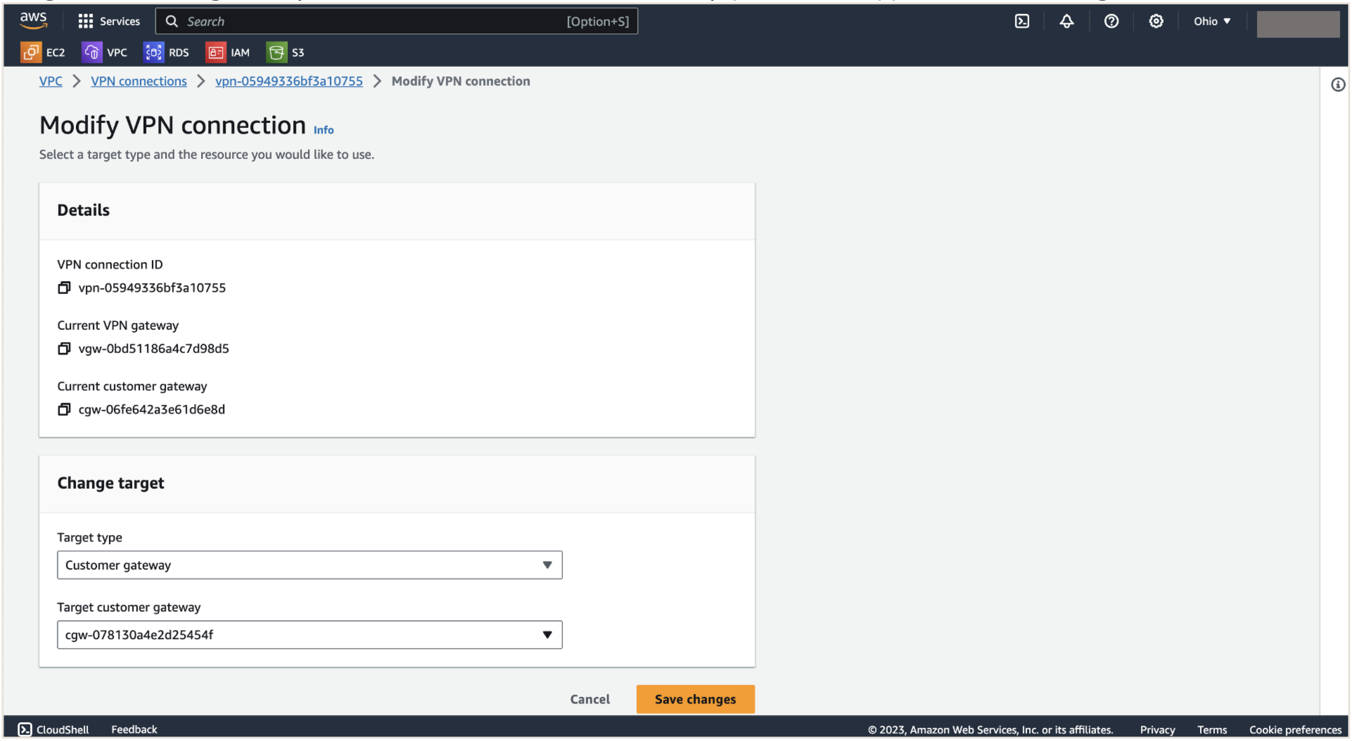
61. Enter a **customer gateway name**. For **BGP ASN**, enter **31898** and for **IP address**, enter the **Oracle VPN IP address for tunnel 1**. Leave everything as-is and click **Create customer gateway**.



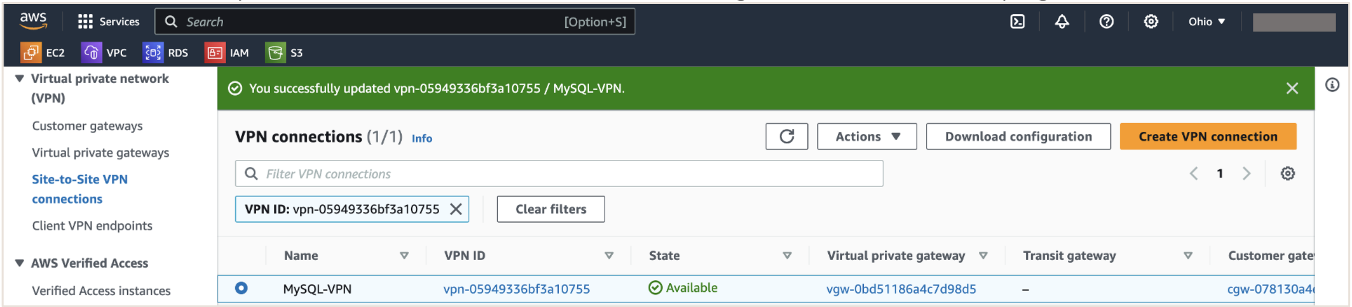
62. From the left-hand AWS menu, scroll down and click **Site-to-Site VPN Connections** under Virtual Private Network (VPN). Select your VPN connection and click the **Actions** button, then **Modify VPN connection**.



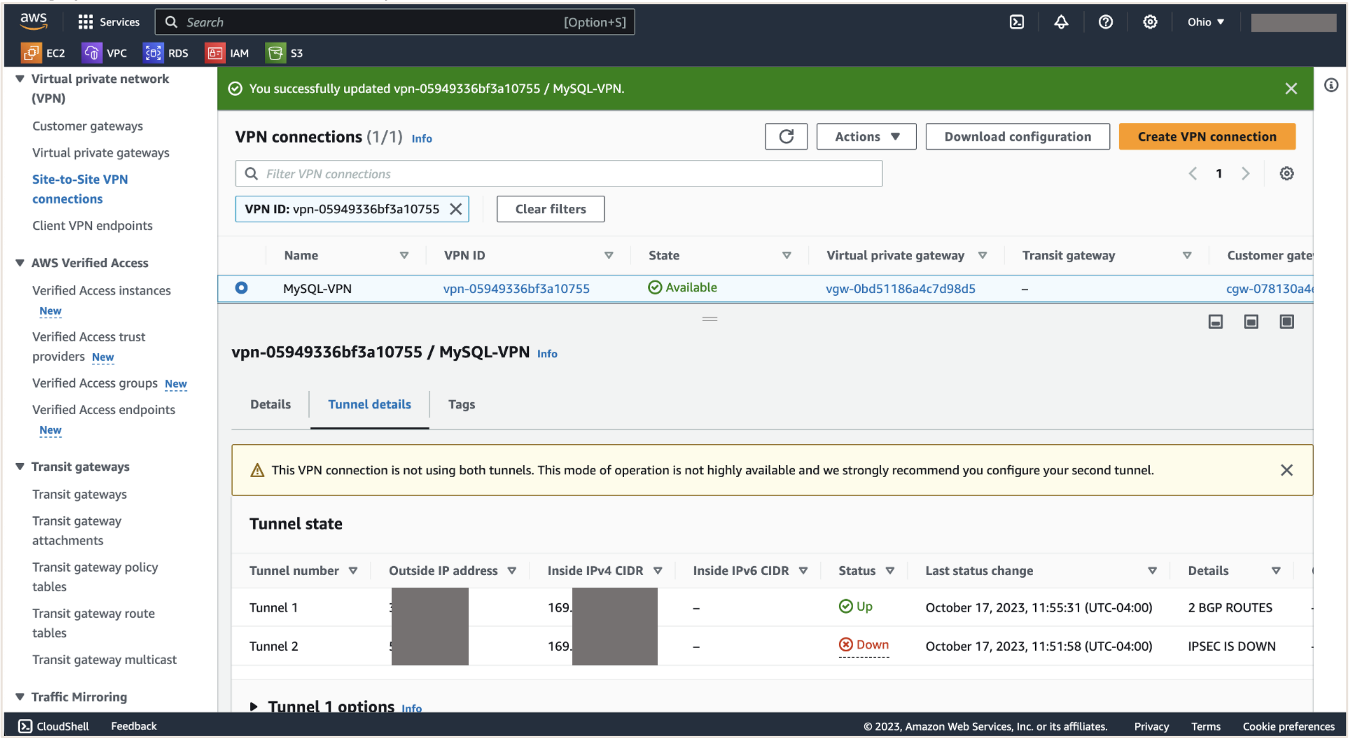
63. You will land on the Modify VPN connection page. Under **Target type**, select **Customer gateway** and for **Target customer gateway**, select the **new Customer Gateway** (not the Temp). Click **Save changes**.



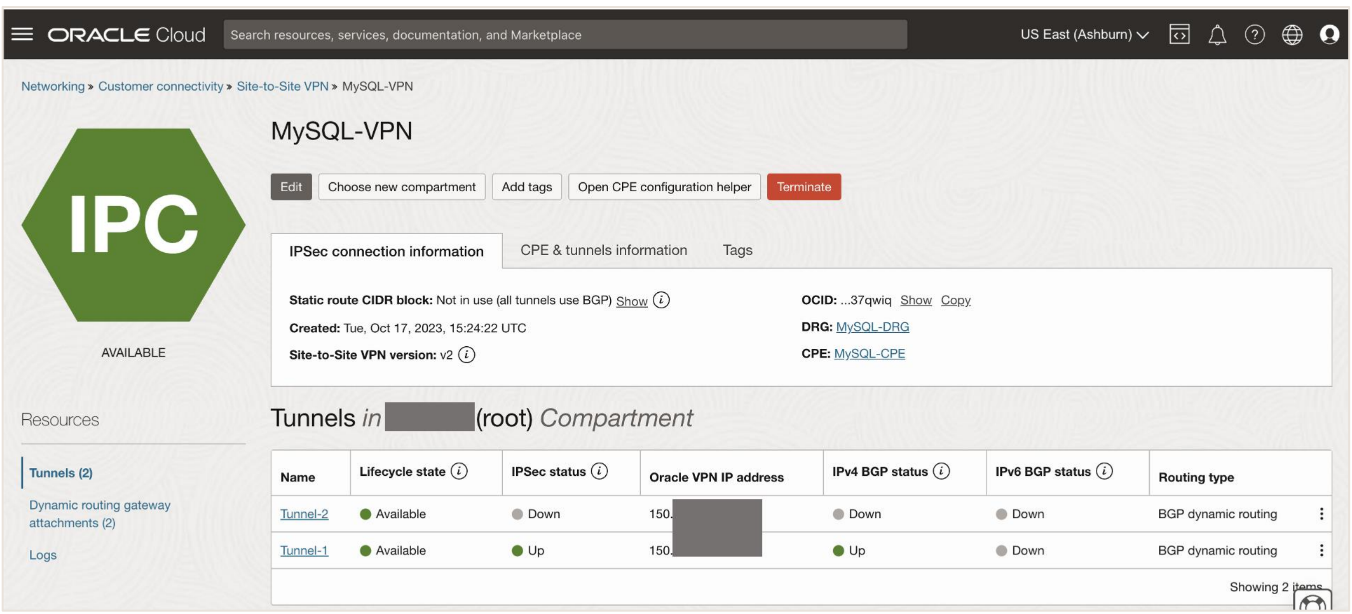
64. After a few minutes, your modified VPN connection should change its **State** from Modifying to **Available**.



65. The VPN connection from OCI to AWS is now setup. To verify if your VPN tunnel is up, select your VPN connection and go to the **Tunnel details** tab which can be found on the same page. You should see a **Status** of **Up** (this will take a few minutes).



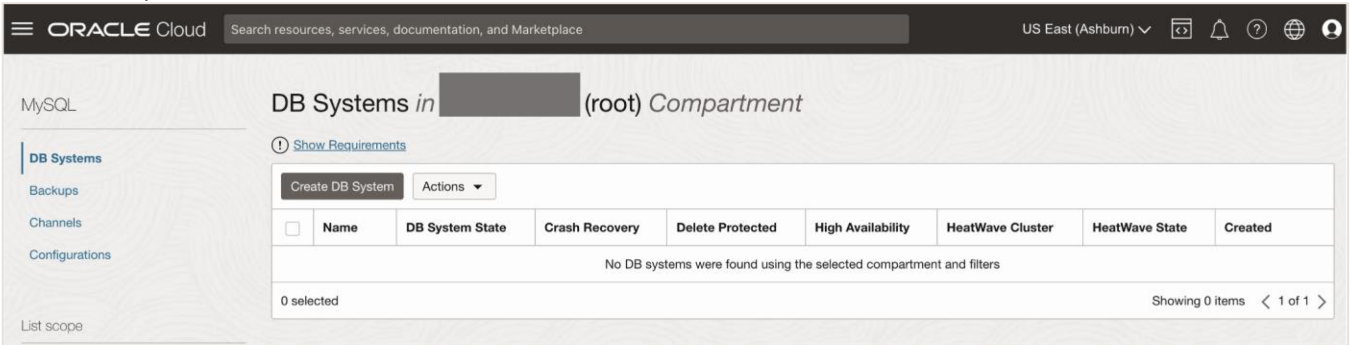
66. You can verify the same on the OCI side. Select your Site-to-Site VPN and under the Resources, click **Tunnels** (the page where you got the Oracle VPN IP address). You should see an **Up** status for **IPSec status** and **IPv4 BGP status**.



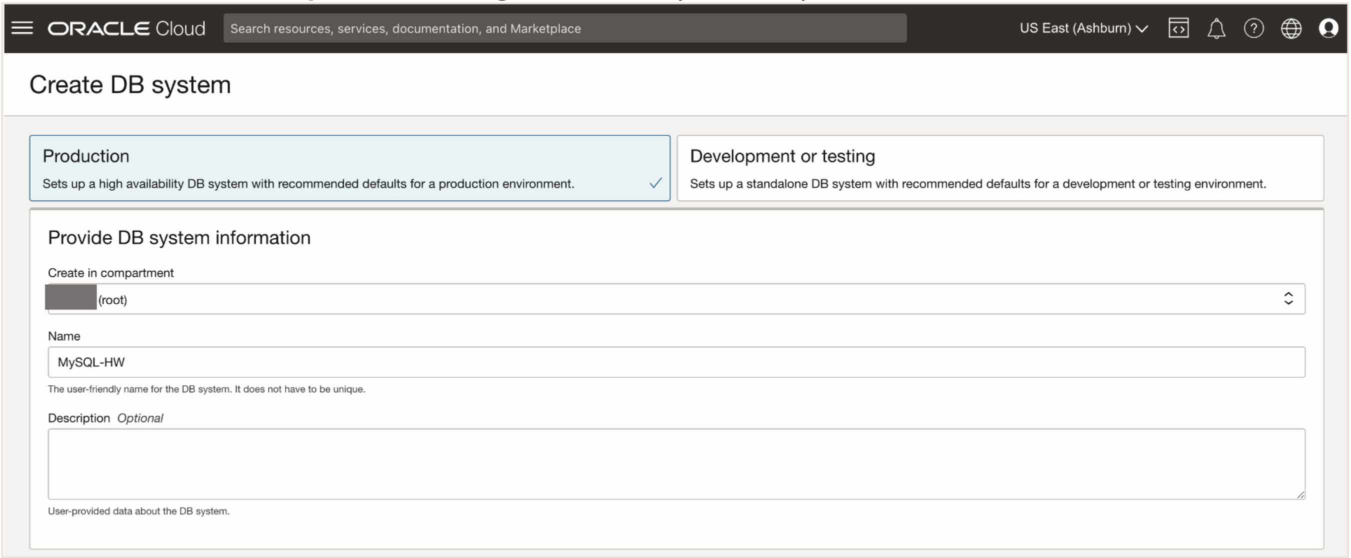
67. We are now ready to perform the migration.

III) On OCI, create a HeatWave MySQL instance.

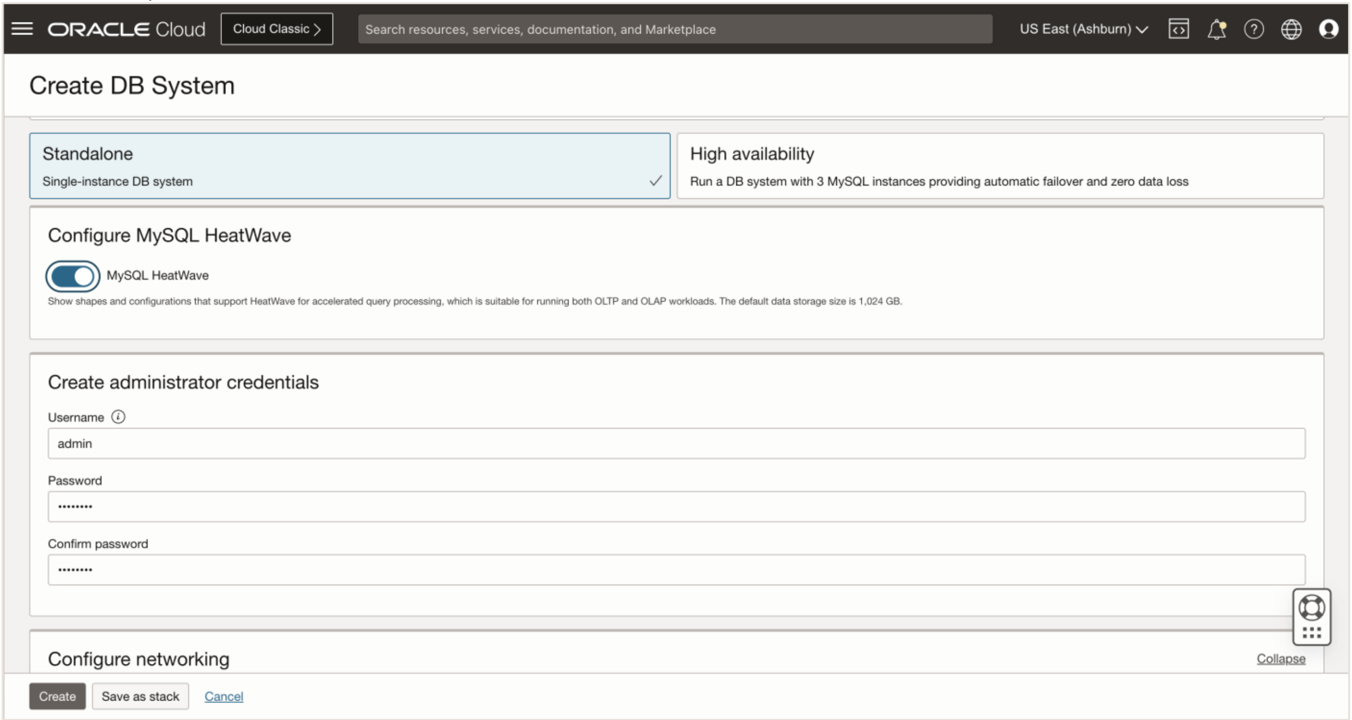
68. From the OCI Console, click on the navigation menu, click **Databases**, and click **HeatWave MySQL**. Click **Create DB System**.



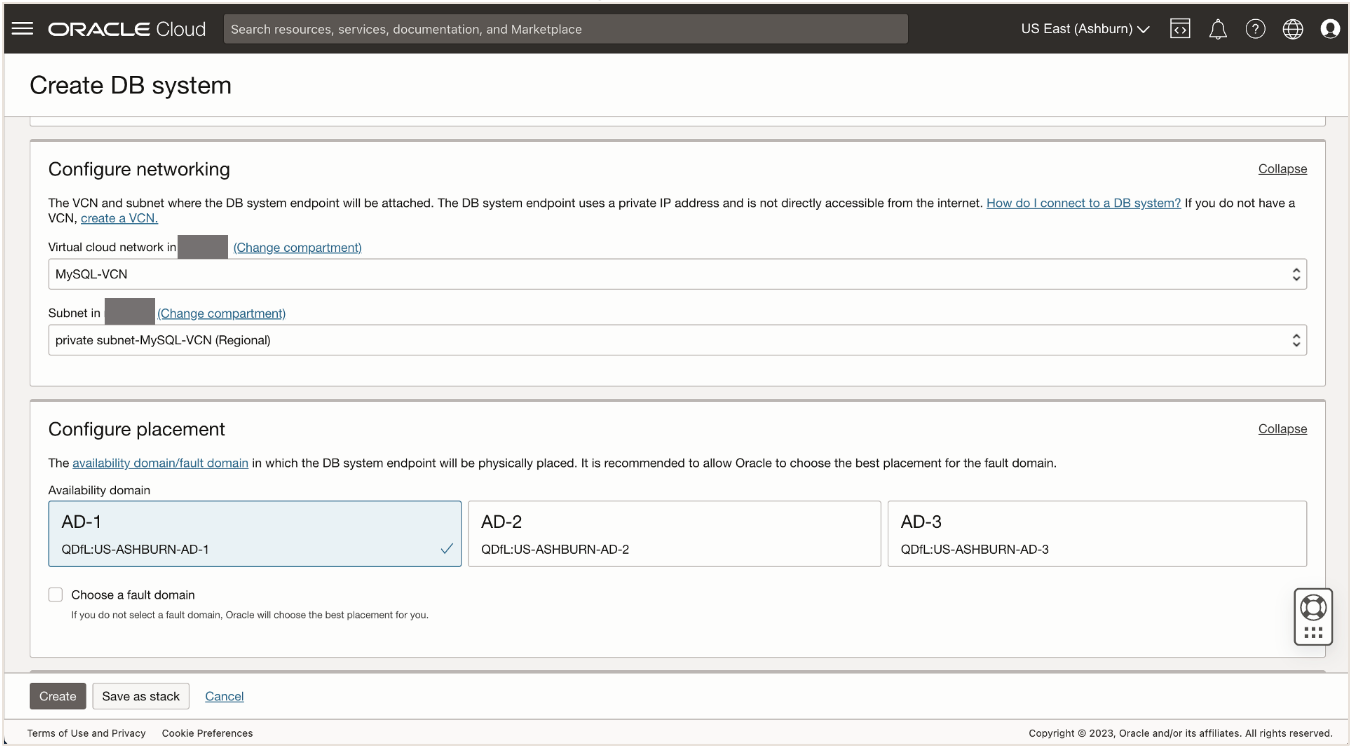
69. Pick **Production** or **Development or testing** and enter **a MySQL DB system name**.



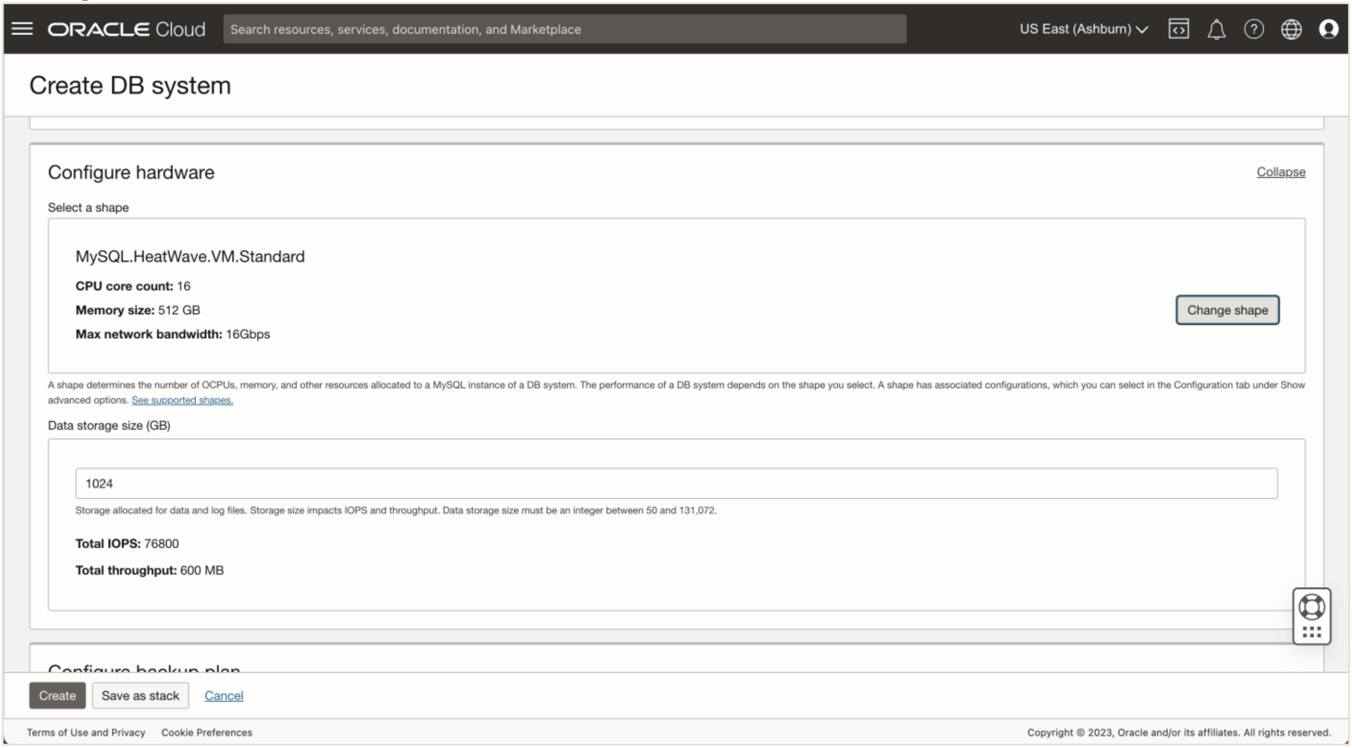
70. Select **Standalone** or **High Availability**. Turn **ON** the button for HeatWave MySQL - if you want to run OLTP, OLAP, and ML workloads. Afterwards, create your **Administrator credentials** that will be used to manage the HeatWave MySQL database.



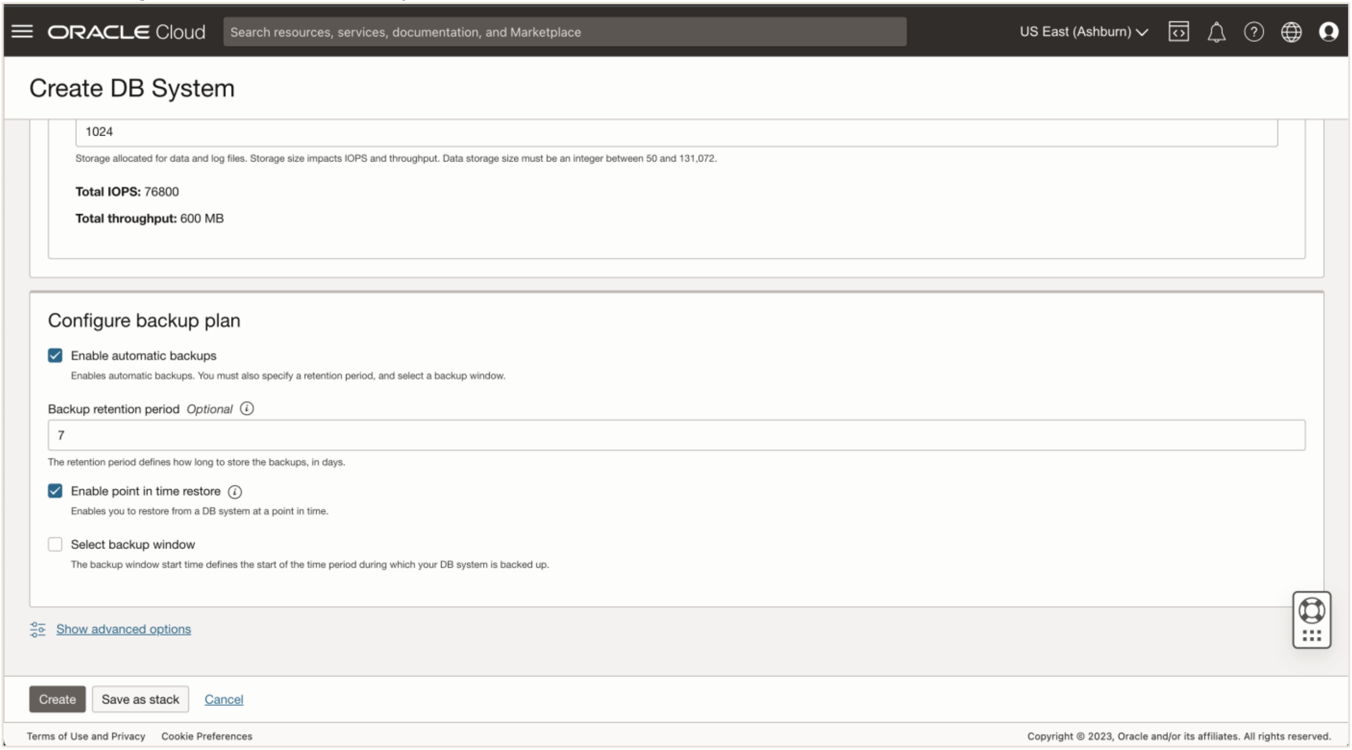
71. For **Configuring Networking** - choose the earlier created VCN and make sure the **Private Subnet** is selected under **Subnet in <compartment-name>**. For **Configure Placement** leave it as-is.

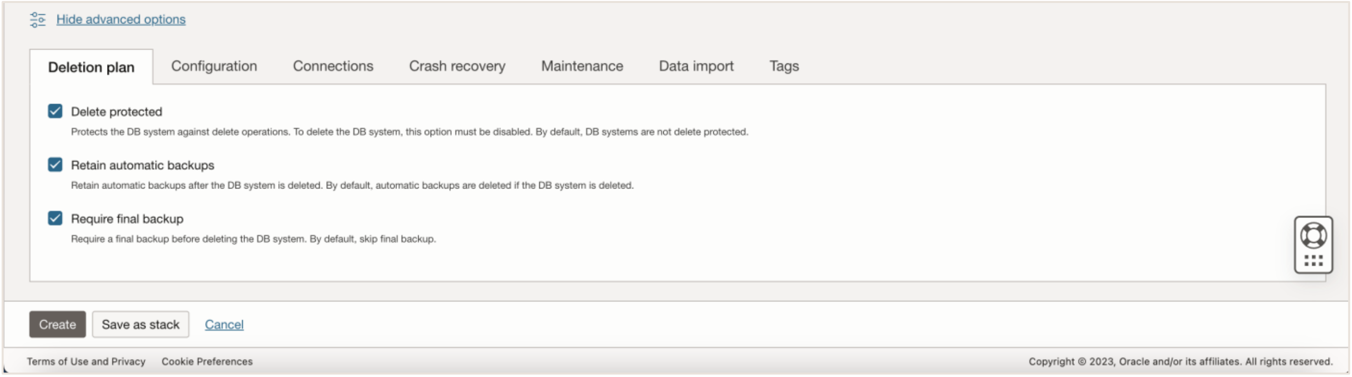


72. **Configure hardware** (OCPU and Memory) for MySQL by choosing an appropriate DB Shape. For this guide, we will use the default HeatWave shape. For the **Data Storage Size** be sure to make the size large enough for future growth.

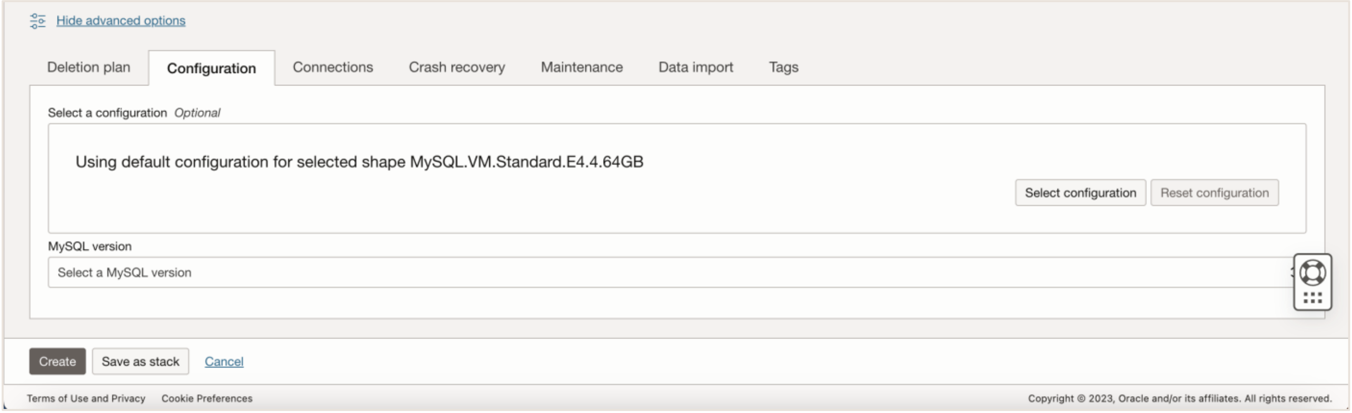


73. **Configure a backup plan** according to what suits your needs. Lastly, scroll down until you see **Show advanced options**. Click on it to expand.

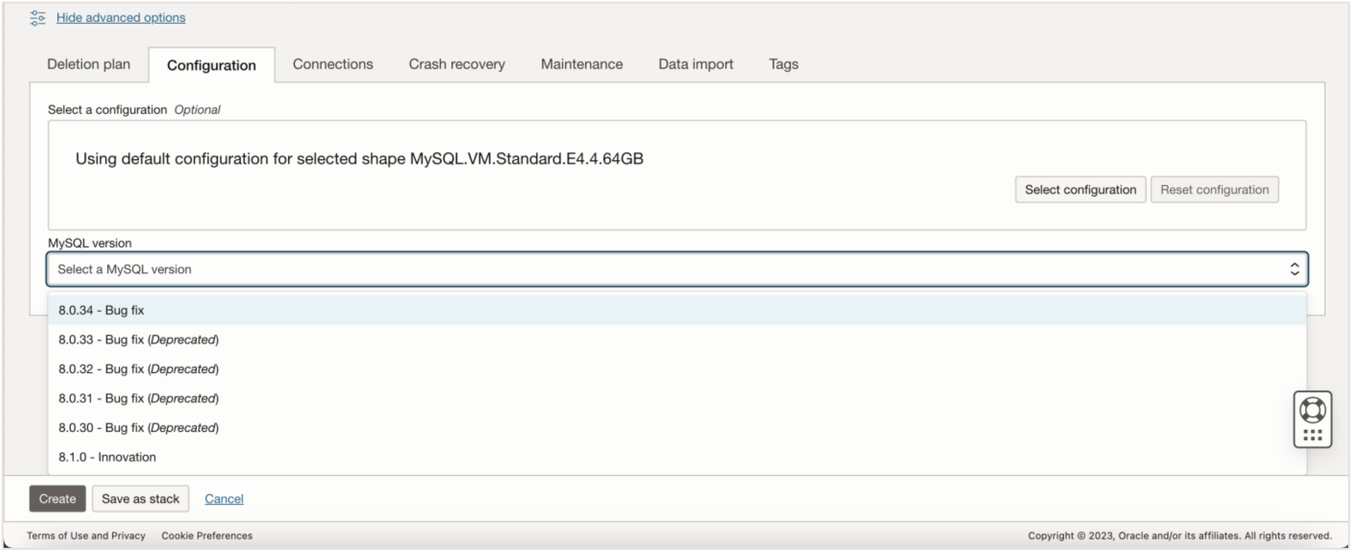




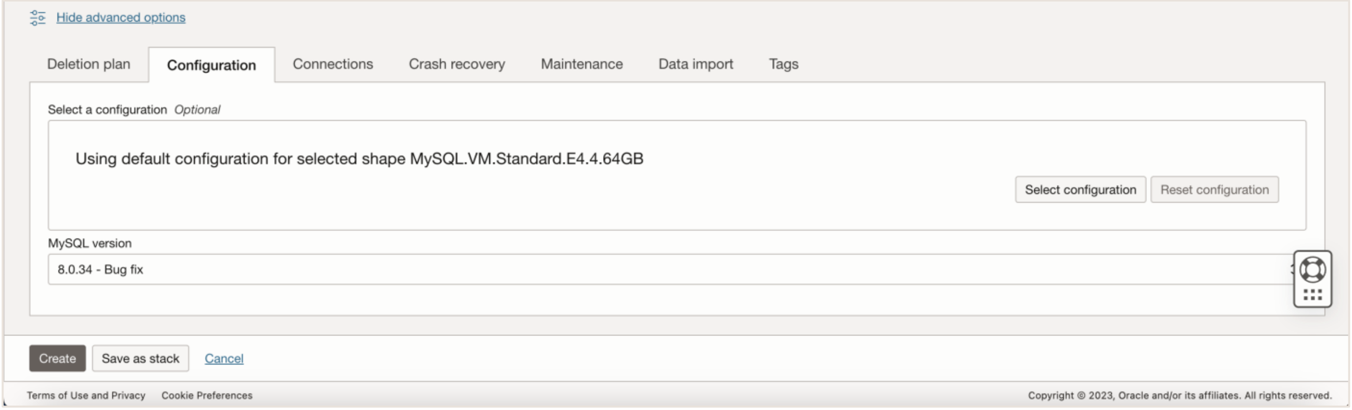
74. From the advanced options screen, go to the **Configuration** tab. If you have a custom configuration that you would like to apply to your HeatWave MySQL instance - you can do so by clicking **Select configuration**. Custom configurations allow you to tweak MySQL variables (i.e., max connections, binary log expire seconds, etc.) rather than using the default values. You must create a custom configuration in advance before applying. For more information regarding custom configurations, see [Configuration of a DB System](). For this guide, we have chosen the default configuration.



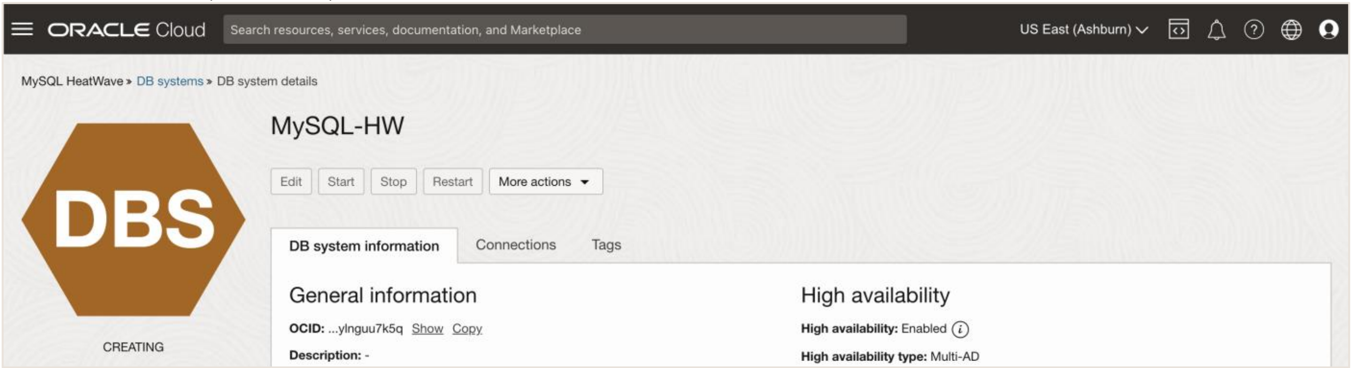
75. For **MySQL version**, choose either **Innovation** or **Bug fix**. With the new MySQL versioning model, you have the flexibility to select an innovation or a bug fix release. Both releases are production-grade quality. MySQL innovation releases allows you to access the latest features and improvements. Innovation releases are ideal for fast-paced development environments with high levels of automated tests and modern continuous integration techniques for faster upgrade cycles. MySQL bug fix releases (aka long-term support releases) allow you to reduce the risks associated with changes in the database software's behavior, as these releases only contain necessary fixes (bugfix and security patches). For more information regarding MySQL innovation and bug fix releases, see [Introducing MySQL Innovation and Bug fix versions](). For this guide, we have chosen **8.0.34 - Bug fix**.



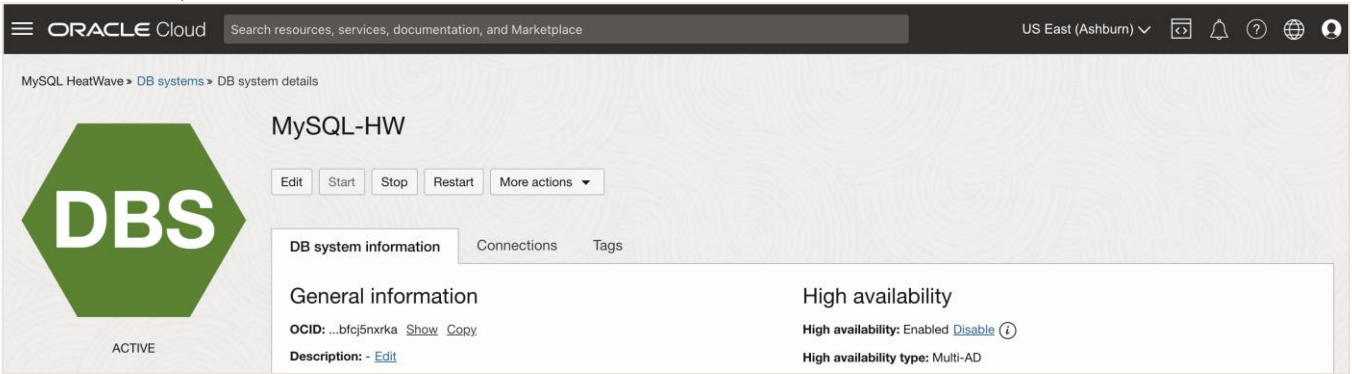
76. Click **Create** to finish the HeatWave MySQL DB system creation process.



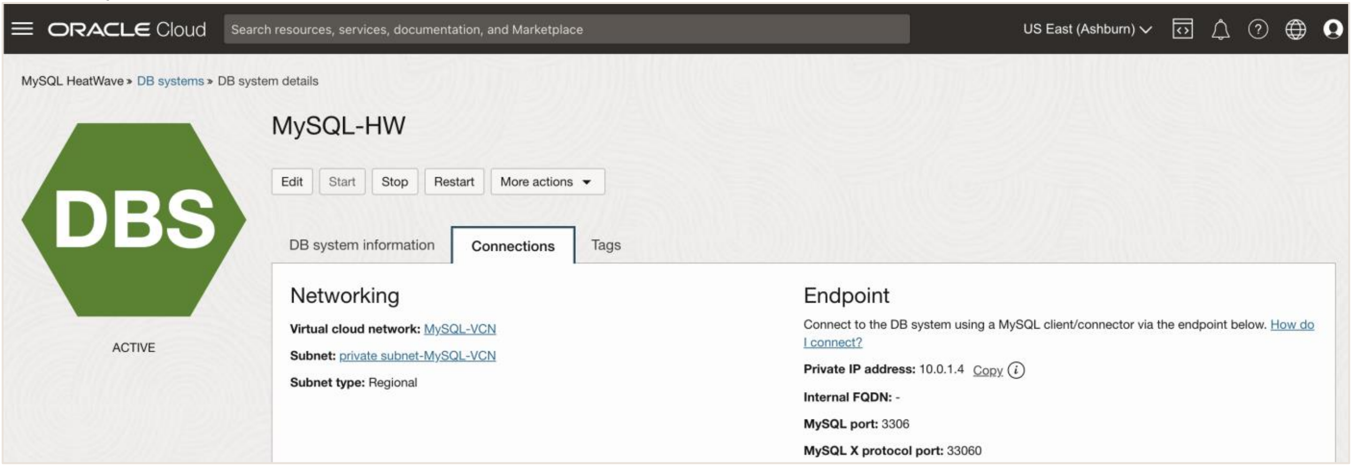
77. Your HeatWave MySQL DB system will start **CREATING**.



78. Within a few minutes, HeatWave MySQL DB system will change its state from CREATING to **ACTIVE** once the instance is ready.



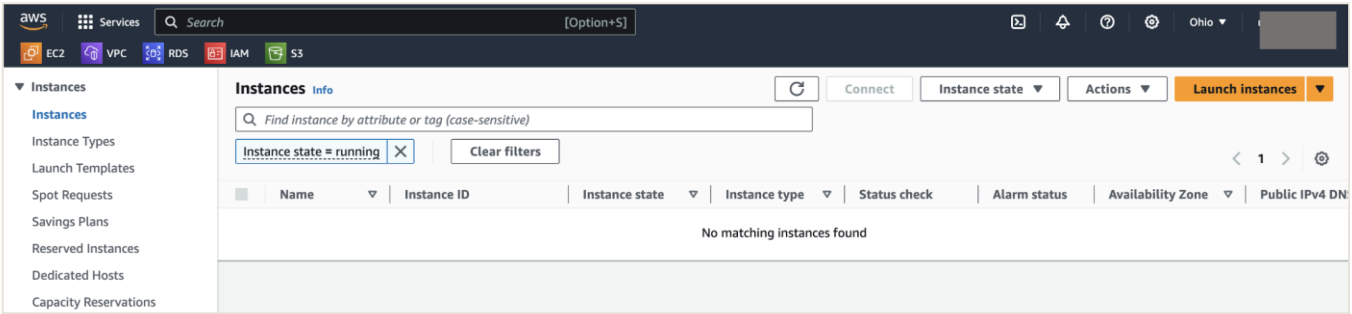
79. On the same DB system details page, click **Connections** to grab the **private IP address** for HeatWave MySQL. Save the private IP Address for later use.



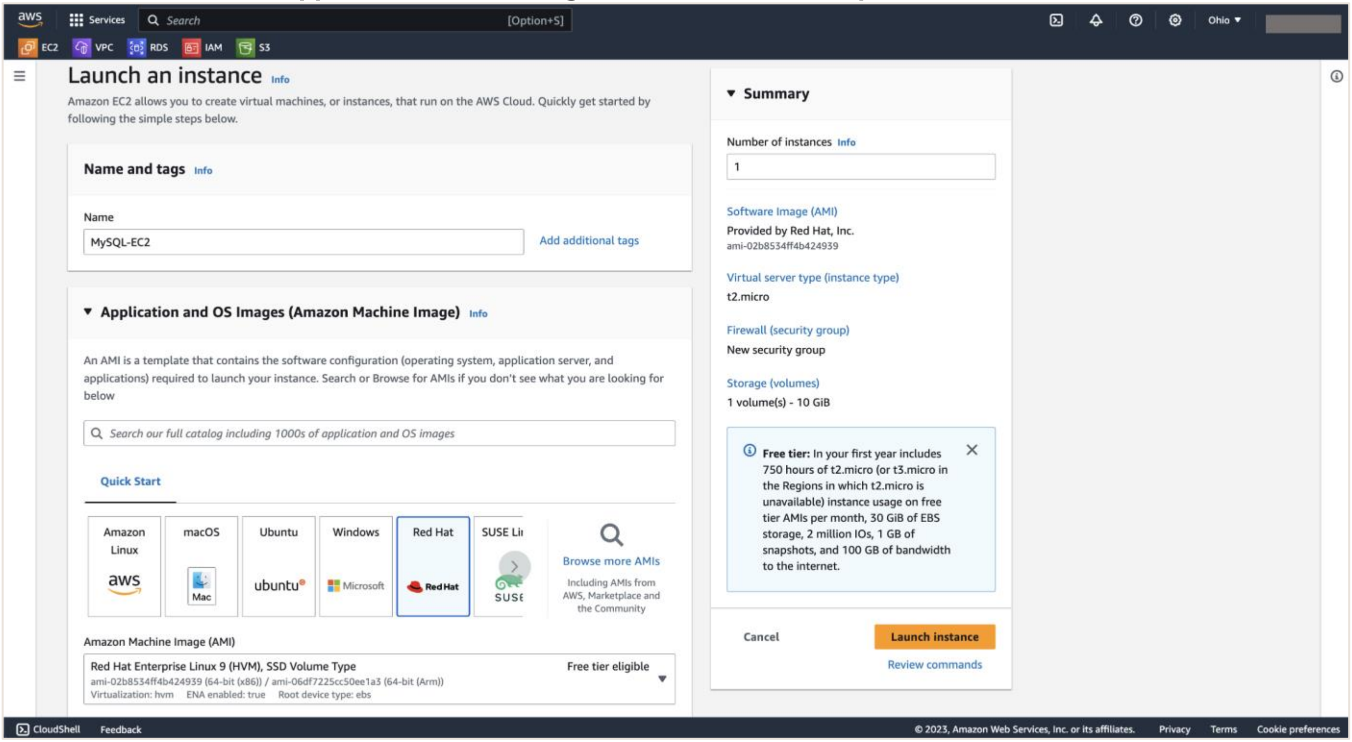
Note: you can navigate to the **DB System Details** page by going to the Navigation menu in OCI. Click **Databases** and click **HeatWave MySQL**. Click on the name of your MySQL DB System to open the **DB System Details** page.

IV) Install MySQL Shell 8.1 or above on an EC2 instance that can connect to Amazon Aurora MySQL.

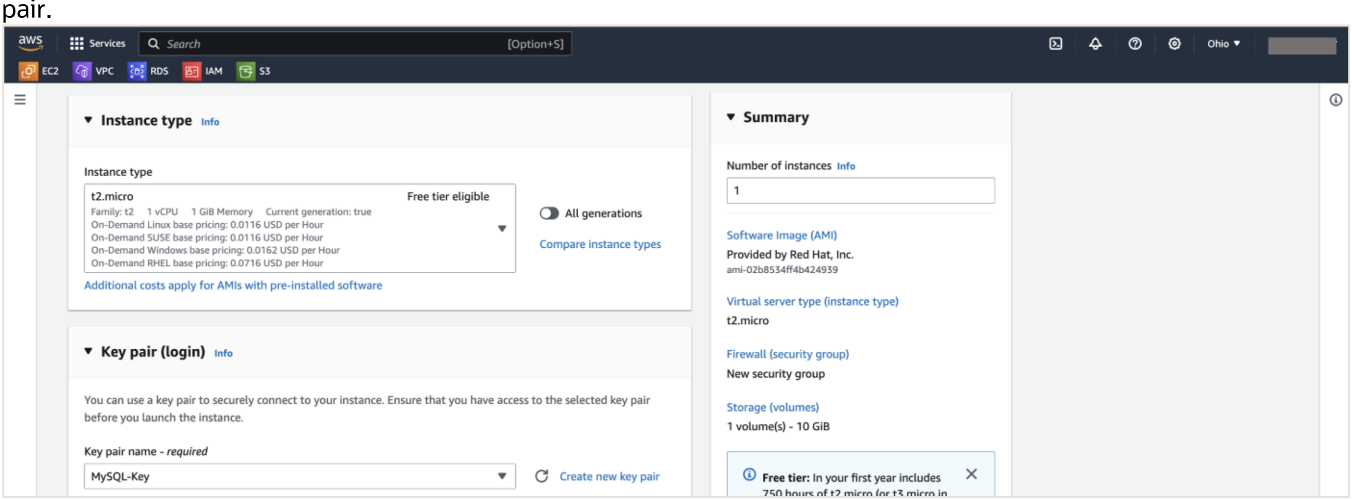
80. Login to [AWS](). From the Services menu, go to **Compute** and select **EC2**.
81. Click **Launch instance**.



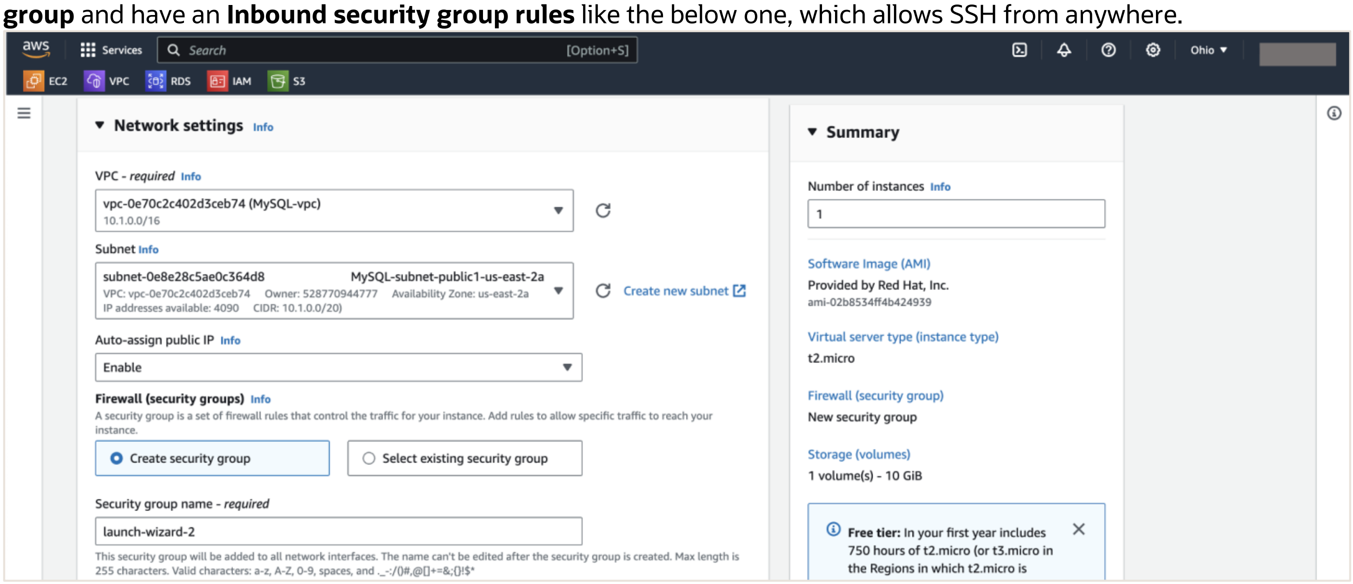
82. Enter an **EC2 name**. For **Application and OS Images**, select **Red Hat Enterprise Linux 9**.

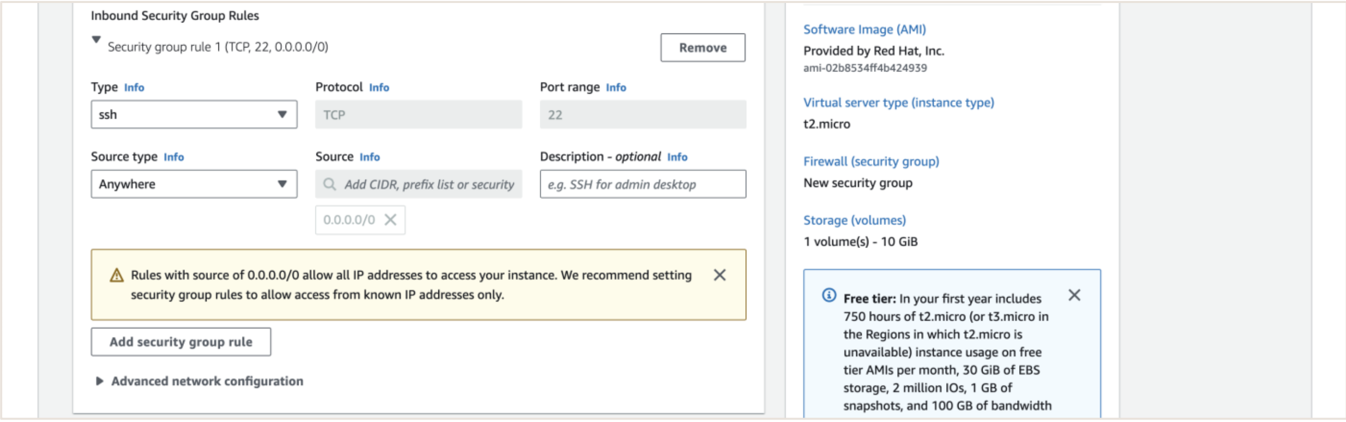


83. For **Instance type**, choose an instance type you think is appropriate. If you have large amounts of data - provisioning an EC2 instance with more vCPUs and Memory will speed up the migration process. For the **Key pair** section, you can use your existing keys or create a new pair. For this guide, we will use an existing key pair.

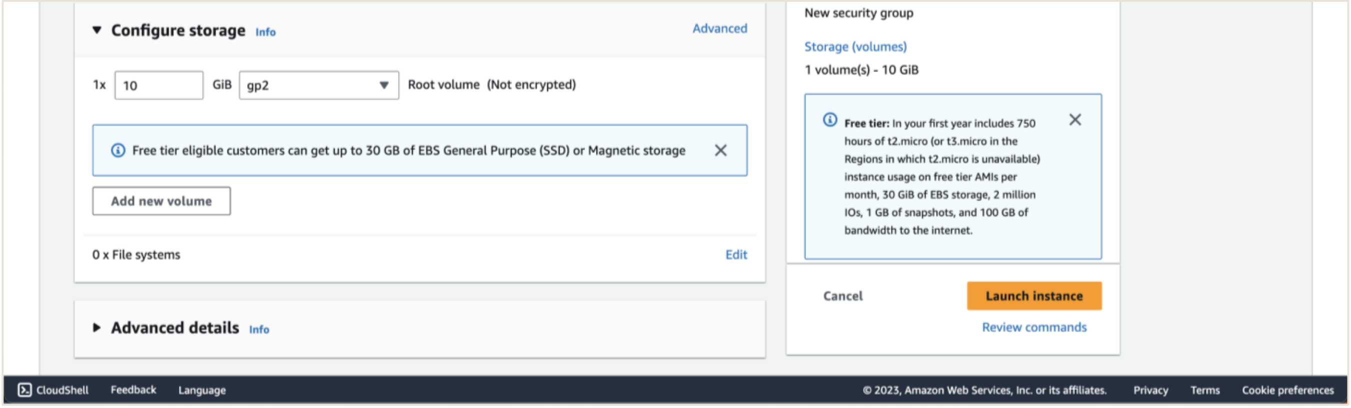


84. Under Network settings, ensure that the correct **VPC** (the VPC that is associated with your Aurora instance) and **Subnet** are selected. For this guide - we have decided to deploy the EC2 instance inside a public subnet. For **Auto-assign public IP** select **Enable**. Under the **Firewall (security groups)**, choose **Create security group** and have an **Inbound security group rules** like the below one, which allows SSH from anywhere.

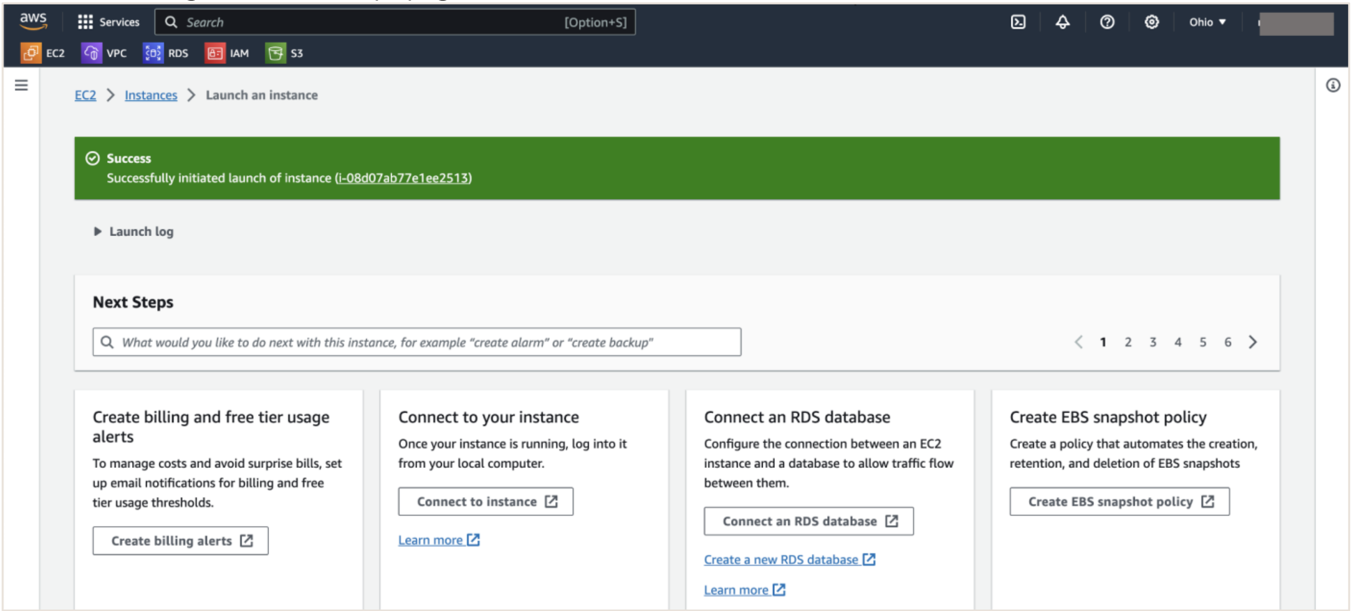




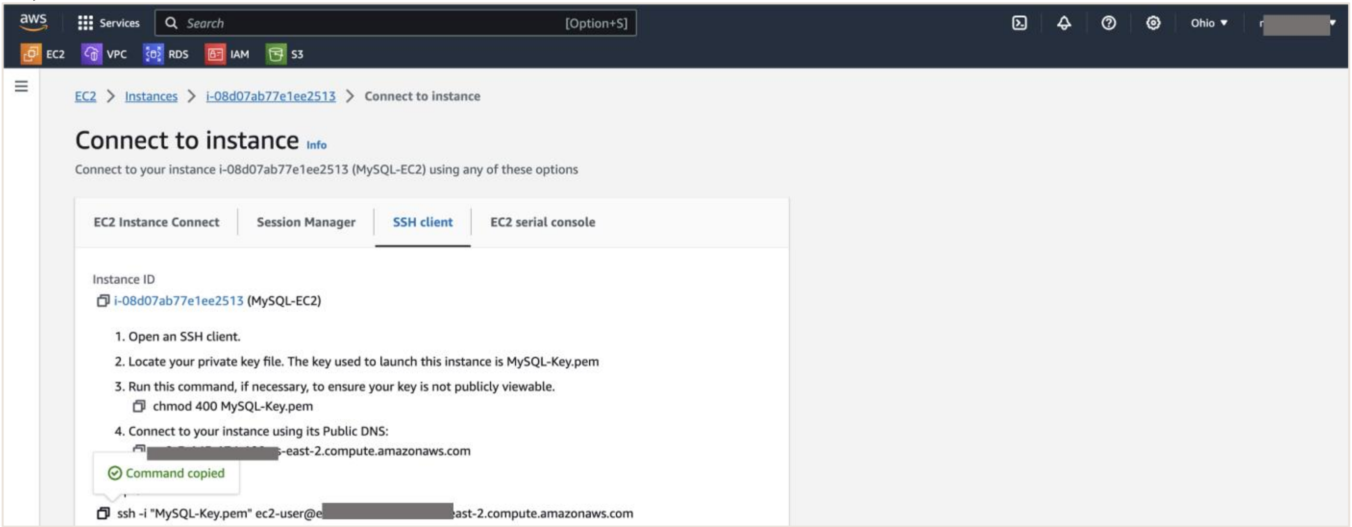
85. Leave everything as-is and click **Launch instance**.



86. You will be brought to a Next Steps page. Here, click **Connect to instance**.



87. If you are using the SSH client to connect to your EC2 instance, copy the **Example** SSH command and login to your EC2 instance.

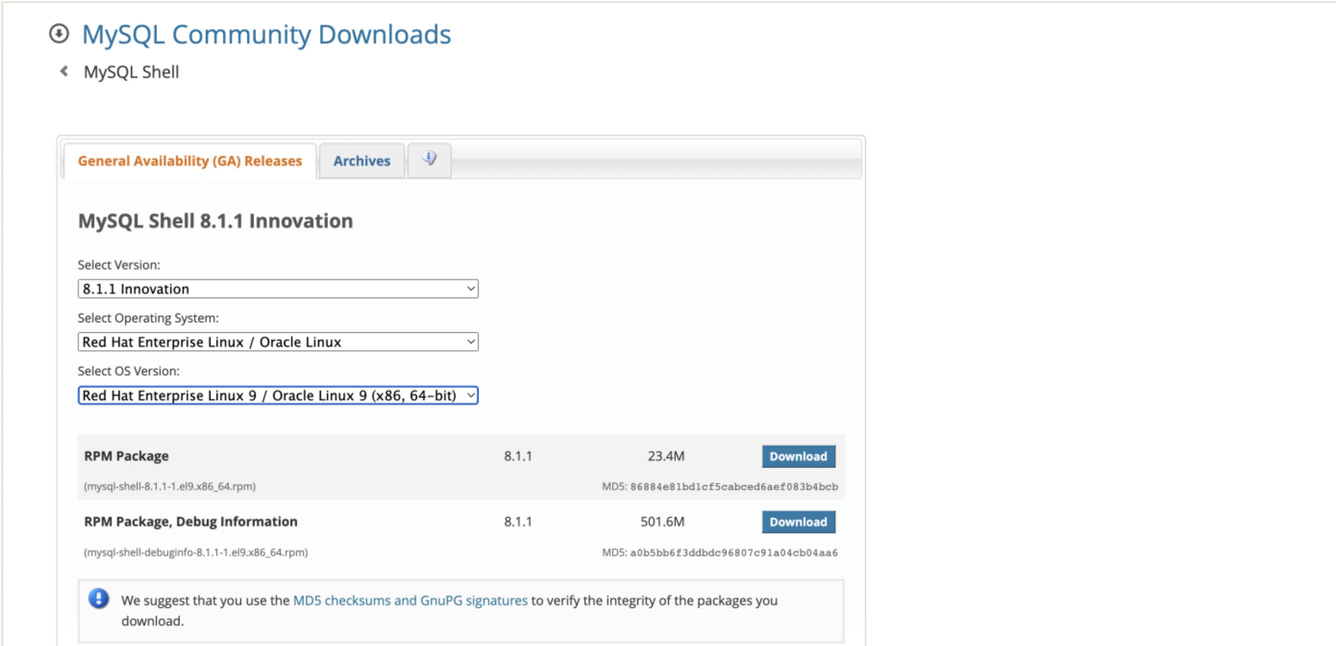


88. You can SSH into EC2 using the below command:
`$ ssh -i </path/to/private-ssh-key> ec2-user@<ec2-Public-DNS>`

Note: after running the above SSH command, if prompted **Are you sure you want to continue connecting (yes/no/[fingerprint])?**, type **yes**.

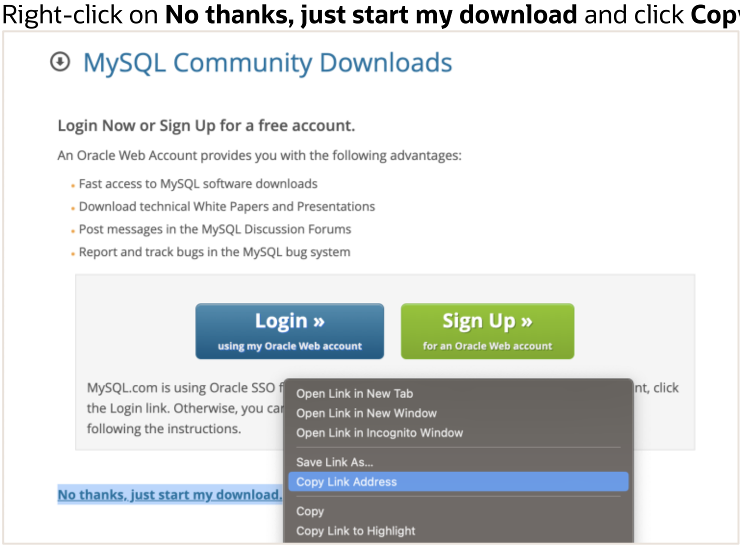
89. We are now successfully connected to the EC2 instance.

90. After making a connection to the EC2 instance, go to the below website and download MySQL Shell 8.1 on your EC2 instance. From the MySQL Shell download page, ensure **8.1.x Innovation or higher** is selected under **Select Version**. MySQL Shell 8.1 is fully compatible with MySQL 8.1, 8.0, and 5.7. For **Operating System** and **OS Version** - pick the appropriate option depending on the OS and the OS Version that you are running. Click **Download**.
https://dev.mysql.com/downloads/shell/



Note: for this guide, we will show you how to install MySQL Shell on a Linux environment. For other environments, see [Installing MySQL Shell on Windows](), [Installing MySQL Shell on Linux](), and [Installing MySQL Shell on macOS]().

91. Right-click on **No thanks, just start my download** and click **Copy link address**.



92. Go back to the EC2 instance that can connect to your Amazon Aurora MySQL and execute the below command to download MySQL Shell:

```
$ wget <MySQL-Shell-Download-Link>
```

Replace the link with what you have.

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el9.x86_64.rpm
```

```
[ec2-user@[redacted] ~]$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el9.x86_64.rpm
--2023-09-20 15:45:58--  https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el9.x86_64.rpm
Resolving dev.mysql.com (dev.mysql.com)... 23.8.77.254, 2600:1407:b800:488::2e31, 2600:1407:b800:484::2e31
Connecting to dev.mysql.com (dev.mysql.com)|23.8.77.254|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el9.x86_64.rpm [following]
--2023-09-20 15:45:58--  https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el9.x86_64.rpm
Resolving cdn.mysql.com (cdn.mysql.com)... 23.60.78.219, 2600:1407:b800:4ae::1d68, 2600:1407:b800:4b1::1d68
Connecting to cdn.mysql.com (cdn.mysql.com)|23.60.78.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24542157 (23M) [application/x-redhat-package-manager]
Saving to: 'mysql-shell-8.1.1-1.el9.x86_64.rpm'

mysql-shell-8.1.1-1 100%[==================>]  23.40M  28.0MB/s    in 0.8s

2023-09-20 15:46:00 (28.0 MB/s) - 'mysql-shell-8.1.1-1.el9.x86_64.rpm' saved [24542157/24542157]
```

Note: to install `wget` on EC2, execute:

```
$ sudo yum install wget
```

93. After downloading the MySQL Shell rpm, install MySQL Shell:

```
$ sudo yum localinstall mysql-shell*
```

```
[ec2-user@[redacted] ~]$ sudo yum localinstall mysql-shell*
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscript
ion-manager to register.

Last metadata expiration check: 0:00:30 ago on Wed 20 Sep 2023 03:54:28 PM UTC.
Dependencies resolved.
================================================================================
 Package              Architecture  Version              Repository         Size
================================================================================
Installing:
 mysql-shell          x86_64        8.1.1-1.el9          @commandline       23 M

Transaction Summary
================================================================================
Install  1 Package
```

94. You can now verify if MySQL Shell has successfully installed on your EC2 instance by executing the below command:

```
$ mysqlsh -version
```

```
[ec2-user@ip-[redacted] ~]$ mysqlsh --version
mysqlsh   Ver 8.1.1 for Linux on x86_64 - for MySQL 8.1.0 (MySQL Community Serv
er (GPL))
[ec2-user@ip-[redacted] ~]$
```

95. To login to your Amazon Aurora MySQL using MySQL Shell, use the below commands:

    `$ mysqlsh <user>@<hostname>:<port-number>`

    -OR-

    `$ mysqlsh -u <user> -p -h <hostname> -P <port-number>`

```
[ec2-user@[redacted] ~]$ mysqlsh admin@database-1-instance-1.[redacted].us
-east-2.rds.amazonaws.com
MySQL Shell 8.1.1

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'admin@database-1-instance-1.[redacted]us-east-2.rds.a
mazonaws.com'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 22204
Server version: 5.7.12 MySQL Community Server (GPL)
No default schema selected; type \use <schema> to set one.
 MySQL  database-1-instance-1.[redacted].us-east-2  JS >
```

Note: you can interact with MySQL Shell using JavaScript, Python, or SQL mode. The default is JavaScript. To switch between the different modes, execute `/js` for JavaScript, `/py` for Python, and `/sql` for SQL mode inside MySQL Shell. To exit out of MySQL Shell, execute `/q`.

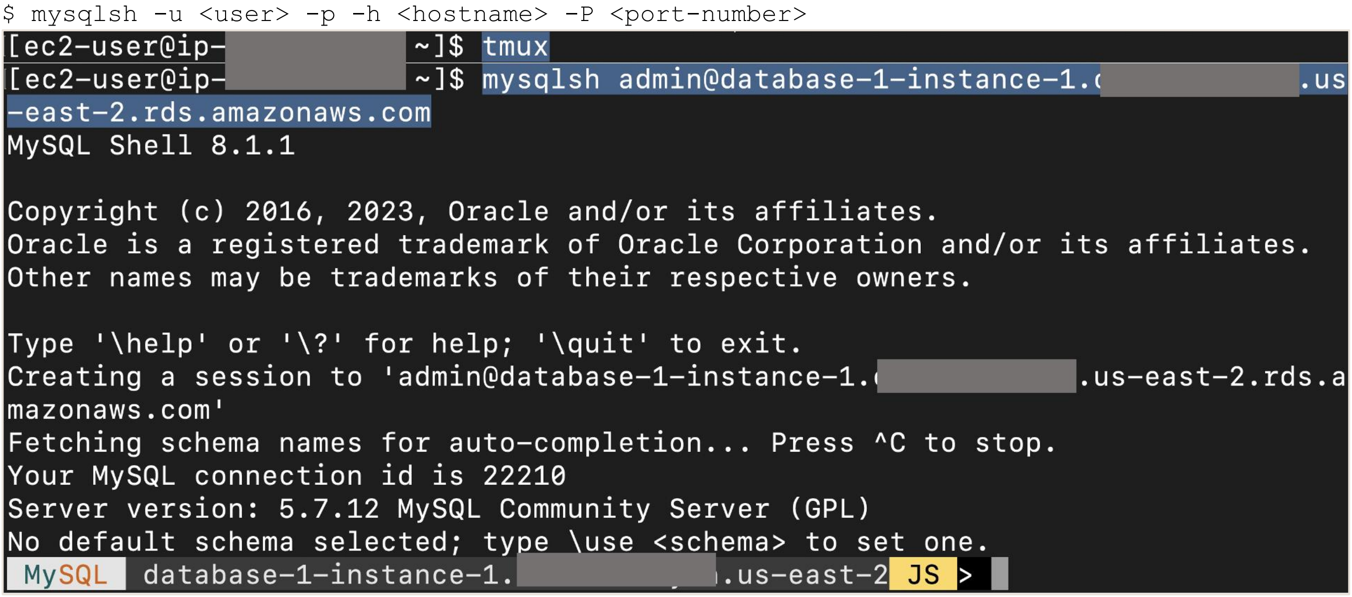
V) Connect to Amazon Aurora MySQL using MySQL Shell on EC2. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from Amazon Aurora MySQL to HeatWave MySQL on OCI.

96. Before connecting to Amazon Aurora MySQL using MySQL Shell and proceeding with the below steps, it is highly recommended that you use a command like **screen** or **tmux**. These commands will allow you to reconnect to a dropped session in case your connection drops in the middle of performing the MySQL Shell export using `util.copyInstance()`. For small databases, the screen or tmux may not be necessary. For this guide, we will use tmux. To learn more about tmux, see [A beginner's guide to tmux](). Below are the basics of using the tmux command:
    - Install tmux on Linux: `$ sudo yum install tmux`
    - Start a new tmux session, from your terminal execute: `$ tmux`
    - List all the active tmux sessions: `$ tmux ls`
    - Detach from a tmux session and leave it running in the background: `$ Ctrl+B d`
    - Attach a tmux session running in the background: `$ tmux attach`
    - End a tmux session: `$ Ctrl+B &`
97. Start a tmux session and connect to your Amazon Aurora MySQL using MySQL Shell on EC2.

```
$ tmux
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```



98. Change to the JavaScript mode of MySQL Shell and run the `util.copyInstance()` utility to export all Amazon Aurora MySQL data into HeatWave MySQL on OCI.

```
MySQL JS> \js
MySQL JS> util.copyInstance('mysql://admin@10.0.1.4', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4,
dryRun:"true"})
```

Note: replace the username (`admin`) and IP address (`10.0.1.4`) with your HeatWave MySQL username and IP address (not the Amazon Aurora MySQL username and IP address).

```
 MySQL  database-1-instance-1.[redacted].us-east-2  JS > util.copyInstance('mysql://adm
in@10.0.1.4', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definer
s", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_inv
alid_grants", "create_invisible_pks"], users: "true", threads: 4, dryRun:"true"})
Please provide the password for 'admin@10.0.1.4': ********
Save password for 'admin@10.0.1.4'? [Y]es/[N]o/Ne[v]er (default No): Y
Copying DDL, Data and Users from in-memory FS, source: ip-[redacted]3306, target: w7gp
26pewjjmqlaw:3306.
SRC: dryRun enabled, no locks will be acquired and no files will be created.
NOTE: SRC: Backup lock is not supported in MySQL 5.7 and DDL changes will not be blocked
. The dump may fail with an error if schema changes are made while dumping.
SRC: Acquiring global read lock
WARNING: SRC: The current user lacks privileges to acquire a global read lock using 'FLU
SH TABLES WITH READ LOCK'. Falling back to LOCK TABLES...
ERROR: SRC: The current user does not have required privileges to execute FLUSH TABLES W
ITH READ LOCK.
    Backup lock is not supported in MySQL 5.7 and DDL changes cannot be blocked.
    The gtid_mode system variable is set to OFF or OFF_PERMISSIVE.
    The log_bin system variable is set to OFF or the current user does not have required
 privileges to execute SHOW MASTER STATUS.
The consistency of the dump cannot be guaranteed.
ERROR: SRC: Unable to acquire global read lock neither table read locks.
SRC: Global read lock has been released
Initializing - done
Util.copyInstance: While 'Initializing': Unable to lock tables: Consistency check has fa
iled. (MYSQLSH 52002)
```

99. Running the above step 98 command may generate **Errors** regarding **table locks** (see step 98 image). If you do encounter such problem (if and only if) run the same command as in step 98 but this time add an additional option: `consistent: "false"` and re-run the command.

```
MySQL JS> util.copyInstance('mysql://admin@10.0.1.4', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4,
dryRun:"true", consistent: "false"})
```

Note: replace the username (`admin`) and IP address (`10.0.1.4`) with your HeatWave MySQL username and IP address (not the Amazon Aurora MySQL username and IP address).

```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS > util.copyInstance('mysql:/
/admin@10.0.1.4', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_def
iners", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip
_invalid_grants", "create_invisible_pks"], users: "true", threads: 4, dryRun:"true", con
sistent: "false"})
Copying DDL, Data and Users from in-memory FS, source: ip[redacted]:3306, target: w7gp
26pewjjmqlaw:3306.
SRC: dryRun enabled, no locks will be acquired and no files will be created.
Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 2 out of 3 users will be dumped.
Gathering information - done
WARNING: SRC: The dumped value of gtid_executed is not guaranteed to be consistent
SRC: Checking for compatibility with MySQL Database Service 8.1.1
NOTE: SRC: MySQL Server 5.7 detected, please consider upgrading to 8.0 first.
SRC: Checking for potential upgrade issues.
SRC: The MySQL server at
SRC: database-1-instance-1.[redacted].us-east-2.rds.amazonaws.com:3306, version
SRC: 5.7.12 - MySQL Community Server (GPL), will now be checked for compatibility
SRC: issues for upgrade to MySQL 8.1.1...
```

[… output truncated]

```
TGT: Starting data load
?% (0 bytes / ?), 0.00 B/s, 0 / 3 tables done
Recreating indexes - done
TGT: Executing common postamble SQL
TGT: No data loaded.
TGT: 0 accounts were loaded
TGT: 0 warnings were reported during the load.

---
Dump_metadata:
  Binlog_file: ''
  Binlog_position: 0
  Executed_GTID_set: ''

 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS >
```

Note:

- `util.copyInstance(connectionData[, options])`: MySQL instance copy utility enables copying of an entire instance to another server. By default, this utility includes all schemas, users, indexes, routines, and triggers. See Copy Utilities.
    - `connectionData`: Defines the connection details for the destination server you want to copy to.
- `compatibility`: Apply the specified requirements for compatibility with HeatWave MySQL for all tables in the dump output, altering the dump files as necessary.
    - `force_innodb`: Change `CREATE TABLE` statements to use the InnoDB storage engine for any tables that do not already use it.
    - `skip_invalid_accounts`: You cannot export a user that has no password defined. This option skips any such users.
    - `strip_definers`: Remove the `DEFINER` clause from views, routines, events, and triggers, so these objects are created with the default definer (the user invoking the schema), and change the `SQL SECURITY` clause for views and routines to specify `INVOKER` instead of `DEFINER`. HeatWave MySQL requires special privileges to create these objects with a definer other than the user loading the schema. If your security model requires that views and routines have more privileges than the account querying or calling them, you must manually modify the schema before loading it.
    - `strip_restricted_grants`: Certain privileges are restricted in the HeatWave MySQL. Privileges such as `RELOAD`, `FILE`, `SUPER`, `BINLOG_ADMIN`, and `SET_USER_ID`. You cannot create users granting these privileges. This option strips these privileges from dumped `GRANT` statements.
    - `strip_tablespaces`: Tablespaces have some restrictions in HeatWave MySQL. If you need tables created in their default tablespaces, this option strips the `TABLESPACE=` option from `CREATE TABLE` statements.
    - `ignore_wildcard_grants`: If enabled, ignores errors from grants on schemas with wildcards, which are interpreted differently in systems where the partial_revokes system variable is enabled.
    - `strip_invalid_grants`: If enabled, strips grant statements that would fail when users are copied, such as grants referring to a specific routine that does not exist.
    - `create_invisible_pks`: Primary keys are required by High Availability and HeatWave. If you intend to export data for use in a highly available DB system or a HeatWave DB system, add primary keys as they are not defined on the tables. This compatibility flag adds invisible primary keys to each table that requires them.
- `users`: Include (`true`) or exclude (`false`) users and their roles and grants in the dump.
- `threads`: (Optional) The number of parallel threads to use to copy chunks of data from the MySQL instance. Each thread has its own connection to the MySQL instance. The default is 4. The copy utilities require twice the number of threads, one thread to copy and one thread to write. If threads is set to N, 2N threads are used.
- `dryRun`: Displays information about the copy with the specified set of options, and about the results of HeatWave MySQL Service compatibility checks but does not proceed with the copy. Setting this option enables you to list out all the compatibility issues before starting the copy.

- `consistent`: Enable (`true`) or disable (`false`) consistent data copies by locking the instance for backup during the copy.

100. Once you have run the command in step 98/99 and did not see any errors in the output (warnings are okay), run the same step 98/99 command but this time change the `dryRun` option to `false`.

```
MySQL JS> util.copyInstance('mysql://admin@10.0.1.4', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4,
dryRun:"false", consistent: "false"})
```

Note: replace the username (`admin`) and IP address (`10.0.1.4`) with your HeatWave MySQL username and IP address (not the Amazon Aurora MySQL username and IP address).

```
MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS > util.copyInstance('mysql://admin@10.0.1.4', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4, dryRun:"false", consistent: "false"})
Copying DDL, Data and Users from in-memory FS, source: ip[redacted]:3306, target: w7gp26pewjjmqlaw:3306.
Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 2 out of 3 users will be dumped.
Gathering information - done
WARNING: SRC: The dumped value of gtid_executed is not guaranteed to be consistent
SRC: Checking for compatibility with MySQL Database Service 8.1.1
```

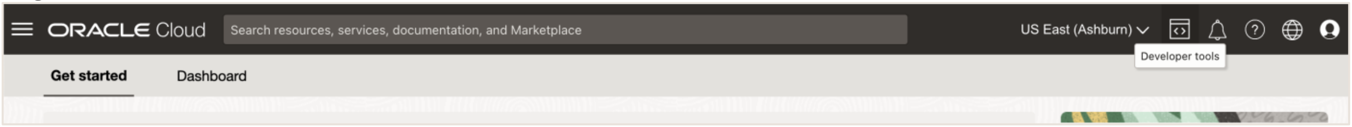
[... output truncated]

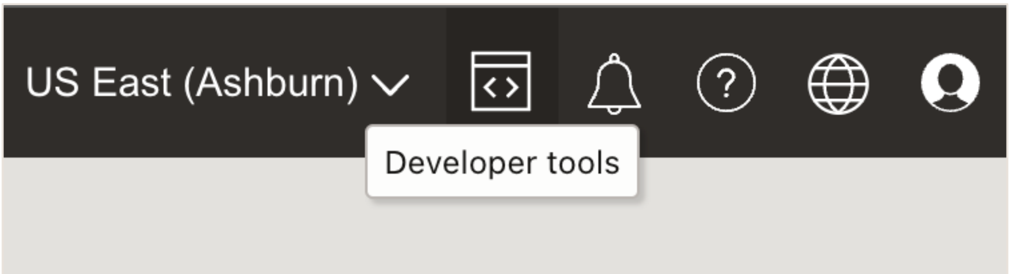
```
100% (5.30K rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s
SRC: Dump duration: 00:00:01s
SRC: Total duration: 00:00:02s
SRC: Schemas dumped: 1
SRC: Tables dumped: 3
SRC: Data size: 194.61 KB
SRC: Rows written: 5302
SRC: Bytes written: 194.61 KB
SRC: Average throughput: 166.71 KB/s
TGT: Executing common postamble SQL
100% (194.61 KB / 194.61 KB), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
TGT: 3 chunks (5.30K rows, 194.61 KB) for 3 tables in 1 schemas were loaded in 1 sec (avg throughput 194.61 KB/s)
TGT: 1 accounts were loaded
TGT: 0 warnings were reported during the load.

---
Dump_metadata:
  Binlog_file: ''
  Binlog_position: 0
  Executed_GTID_set: ''

MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS >
```

Note: once the MySQL Shell copy utility finishes, all your data will be copied over from Amazon Aurora MySQL to HeatWave MySQL on OCI. This completes the migration process. You can end your tmux session.

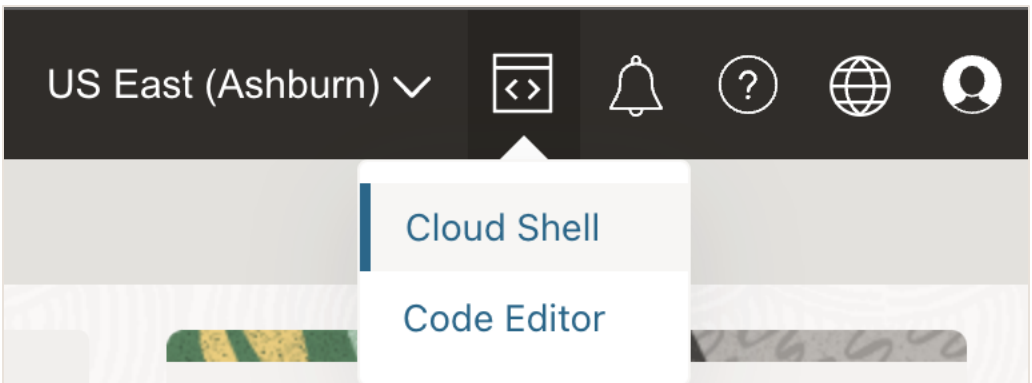
VI) (Optional) On OCI, use the Cloud Shell to verify whether the data was migrated successfully from Amazon Aurora MySQL to HeatWave MySQL on OCI.

101. Login to OCI, navigate to the top right corner and click on **Developer tools** right next to your OCI Region.

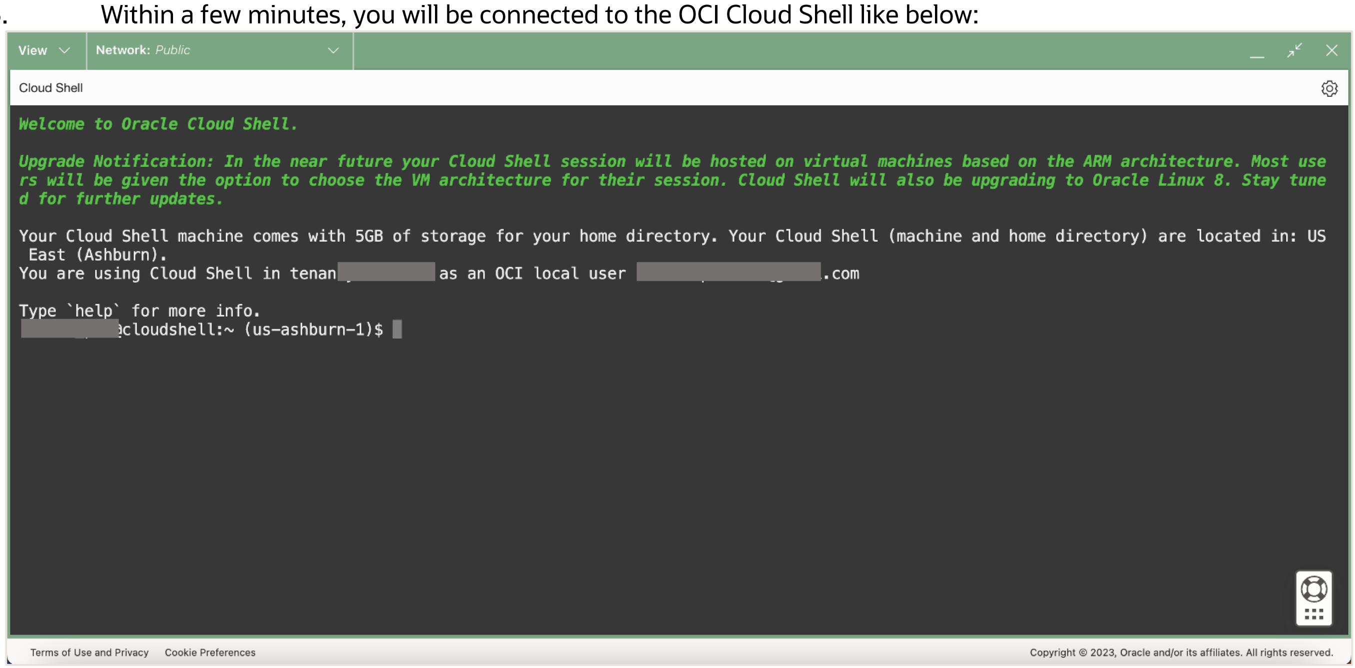




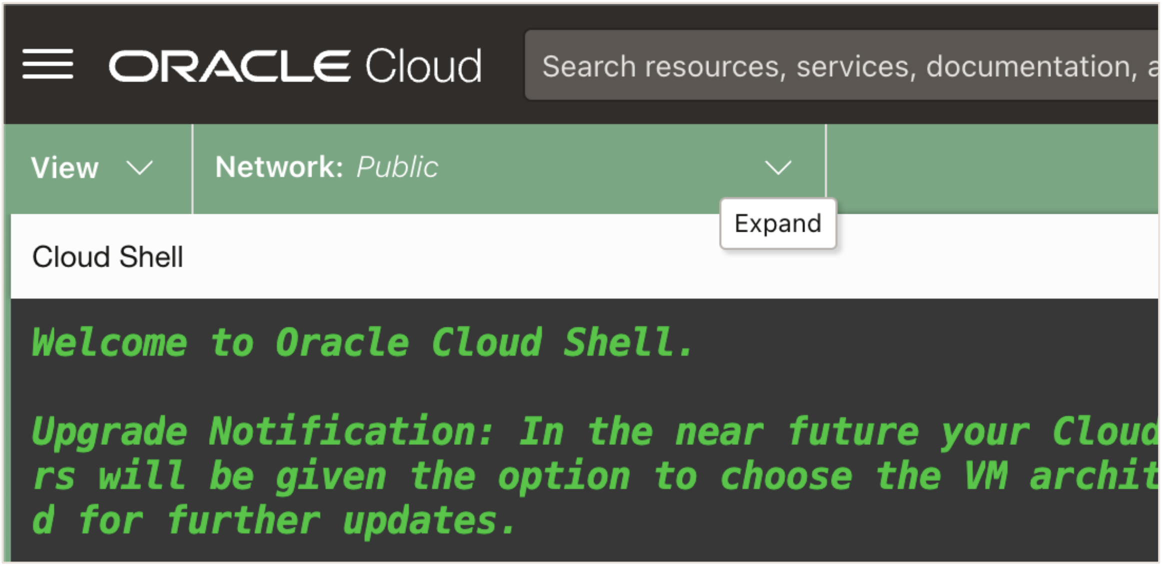
102. Click **Cloud Shell**.

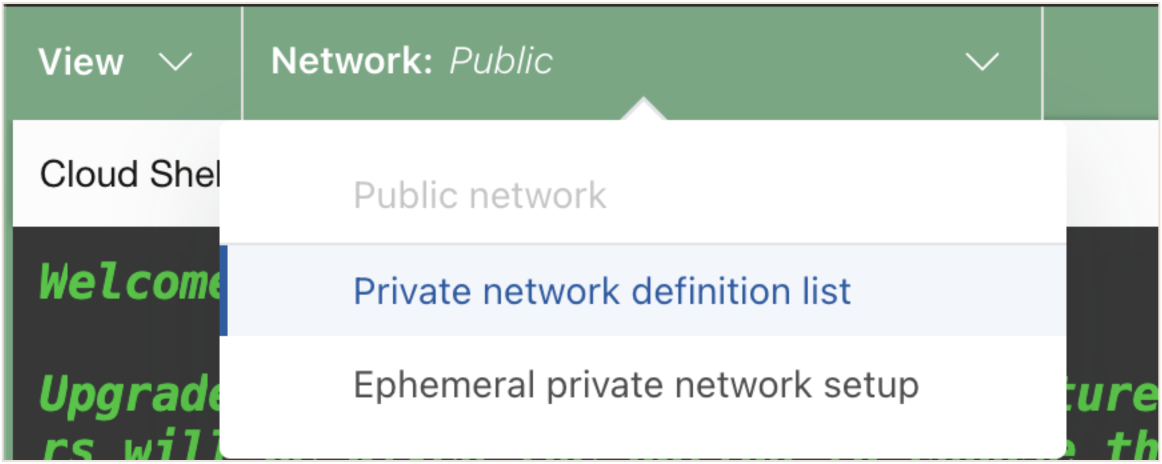


103. Within a few minutes, you will be connected to the OCI Cloud Shell like below:

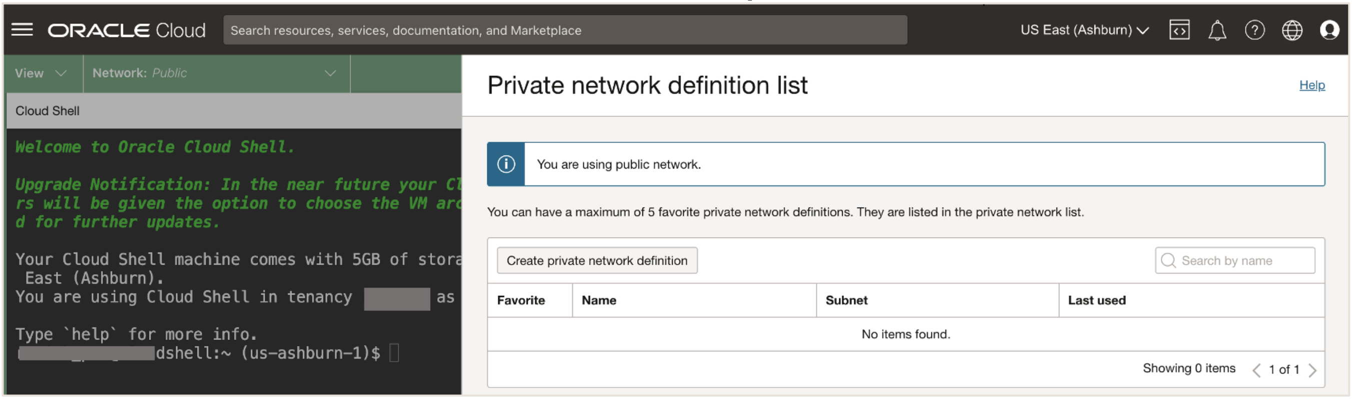


104. Click on the **down arrow** next to **Network: Public** and select **Private network definition list**.

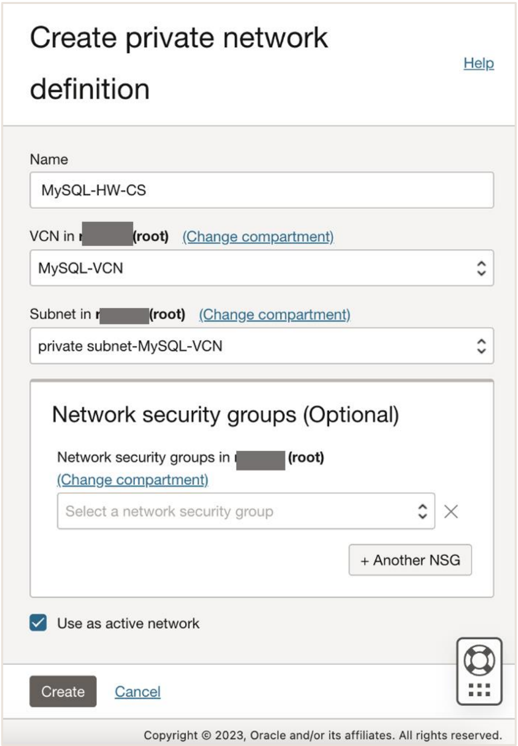




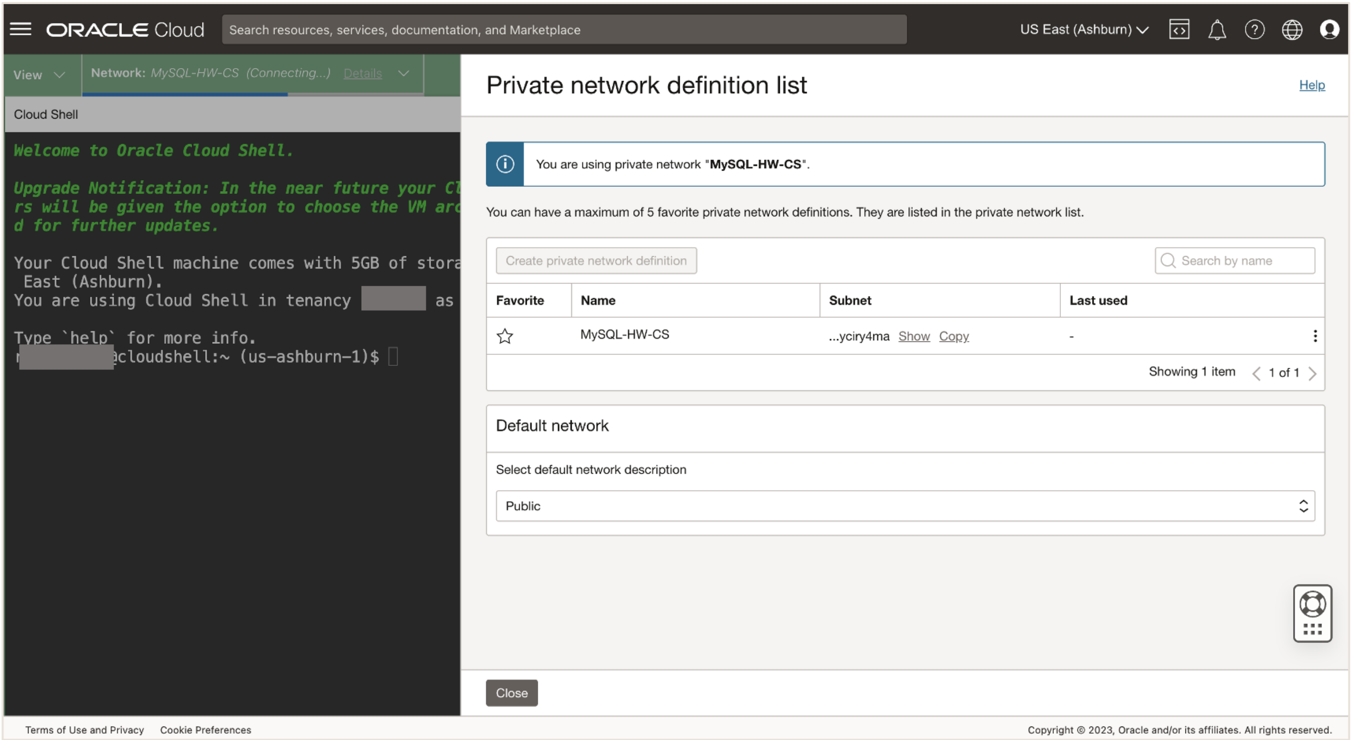
105. On the Private network definition list form, select **Create private network definition**.



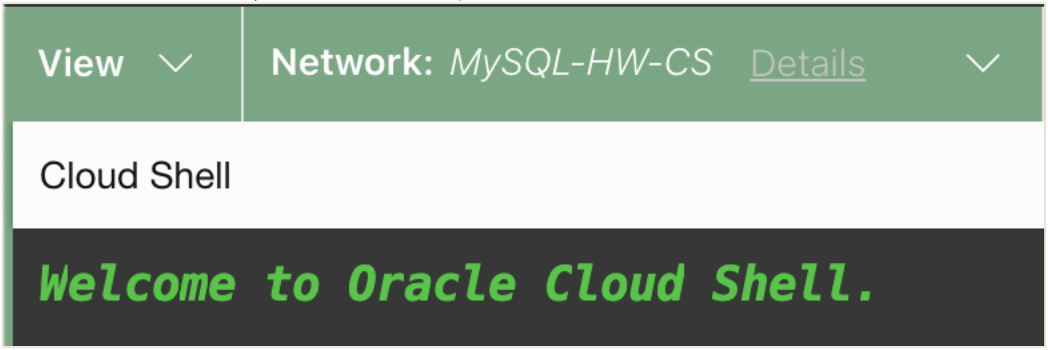
106. Enter a **private network definition name**. From the **VCN in <compartment-name>** dropdown, select the **VCN associated with HeatWave MySQL**. For **Subnet in <compartment-name>** dropdown, select the **private subnet**. Leave the **Network security groups** as-is and **check the box** where it says **Use as active network**. Click **Create**.



107. Click **Close**.



108. Within a few minutes, you will be able to access your private subnet (where HeatWave MySQL resides) from the Cloud Shell. You should see the **Network** change from Public to the **private network definition name** that you entered in step 106.

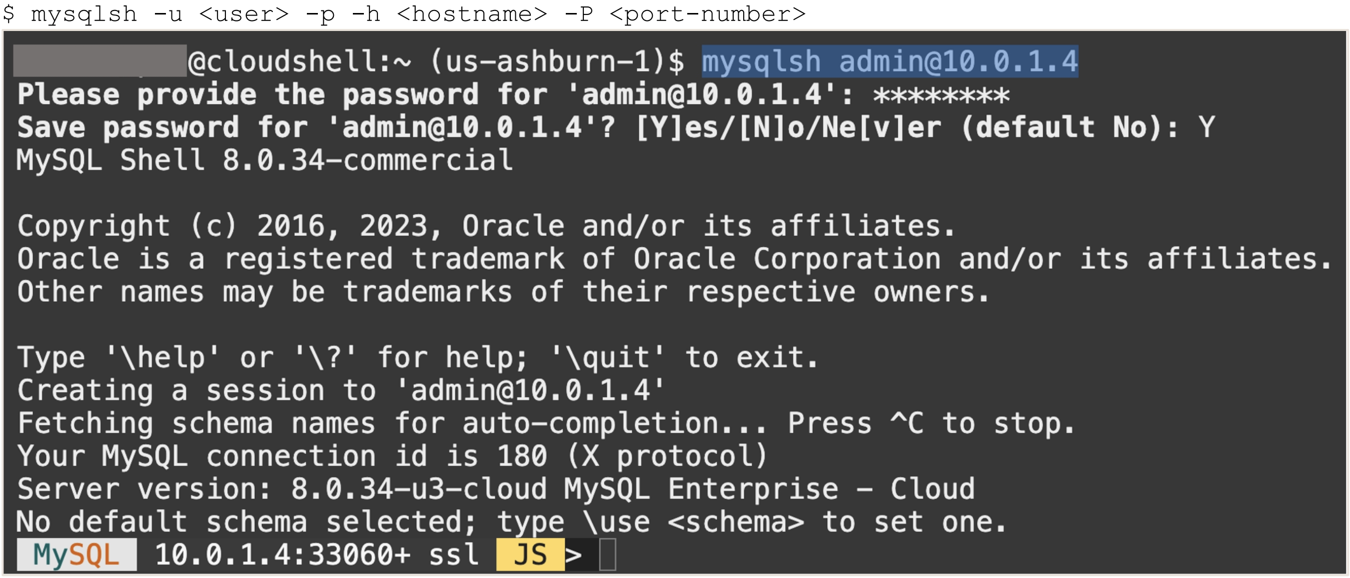


109. From the Cloud Shell terminal, login to your HeatWave MySQL instance (by providing the username and private IP of HeatWave MySQL) using MySQL Shell to validate whether the migration was successful:

```
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```



110. Change to the SQL mode of MySQL Shell and run the below commands:

```
MySQL JS> \sql
MySQL SQL> SHOW SCHEMAS;
MySQL SQL> SHOW TABLES IN <schema-name>;
```

```
 MySQL  10.0.1.4:33060+ ssl  JS > \sql
Switching to SQL mode... Commands end with ;
Fetching global names for auto-completion... Press ^C to stop.
 MySQL  10.0.1.4:33060+ ssl  SQL > SHOW SCHEMAS;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysql_audit        |
| performance_schema |
| sys                |
| world              |
+--------------------+
6 rows in set (0.0009 sec)
 MySQL  10.0.1.4:33060+ ssl  SQL > SHOW TABLES IN world;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.0018 sec)
 MySQL  10.0.1.4:33060+ ssl  SQL >
```

111. You can run the below query on every table that you have for your Amazon Aurora MySQL and HeatWave MySQL on OCI to ensure that the row count matches on both sides:

```
MySQL SQL> SELECT COUNT(*) FROM <schema-name>.<table-name>;
```

112. Here is our row count comparison for Amazon Aurora MySQL and HeatWave MySQL:

Amazon Aurora MySQL row count:

```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds SQL > USE world;
Default schema set to `world`.
Fetching global names, object names from `world` for auto-completion... Press ^C to stop.
 MySQL  database-1-instance-1.[redacted].us-east-2 world  SQL > SELECT COUNT(*) FROM city;
+----------+
| COUNT(*) |
+----------+
|     4079 |
+----------+
1 row in set (0.0019 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2 world  SQL > SELECT COUNT(*) FROM country
;
+----------+
| COUNT(*) |
+----------+
|      239 |
+----------+
1 row in set (0.0008 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2 world  SQL > SELECT COUNT(*) FROM country
language;
+----------+
| COUNT(*) |
+----------+
|      984 |
+----------+
1 row in set (0.0011 sec)
```

HeatWave MySQL row count:

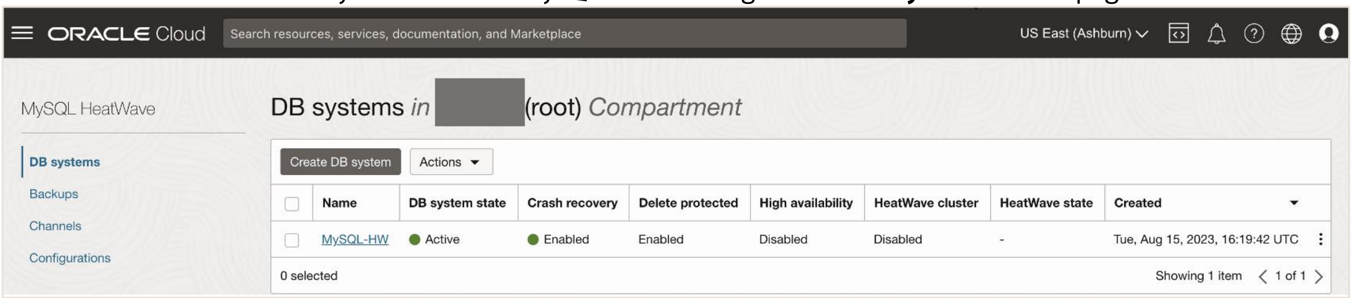
```
 MySQL  10.0.1.4:33060+ ssl  SQL > USE world;
Default schema set to `world`.
Fetching global names, object names from `world` for auto-completion... Press ^C to stop.
 MySQL  10.0.1.4:33060+ ssl  world  SQL > SELECT COUNT(*) FROM city;
+----------+
| COUNT(*) |
+----------+
|     4079 |
+----------+
1 row in set (0.0120 sec)
 MySQL  10.0.1.4:33060+ ssl  world  SQL > SELECT COUNT(*) FROM country;
+----------+
| COUNT(*) |
+----------+
|      239 |
+----------+
1 row in set (0.0050 sec)
 MySQL  10.0.1.4:33060+ ssl  world  SQL > SELECT COUNT(*) FROM countrylanguage;
+----------+
| COUNT(*) |
+----------+
|      984 |
+----------+
1 row in set (0.0086 sec)
 MySQL  10.0.1.4:33060+ ssl  world  SQL >
```

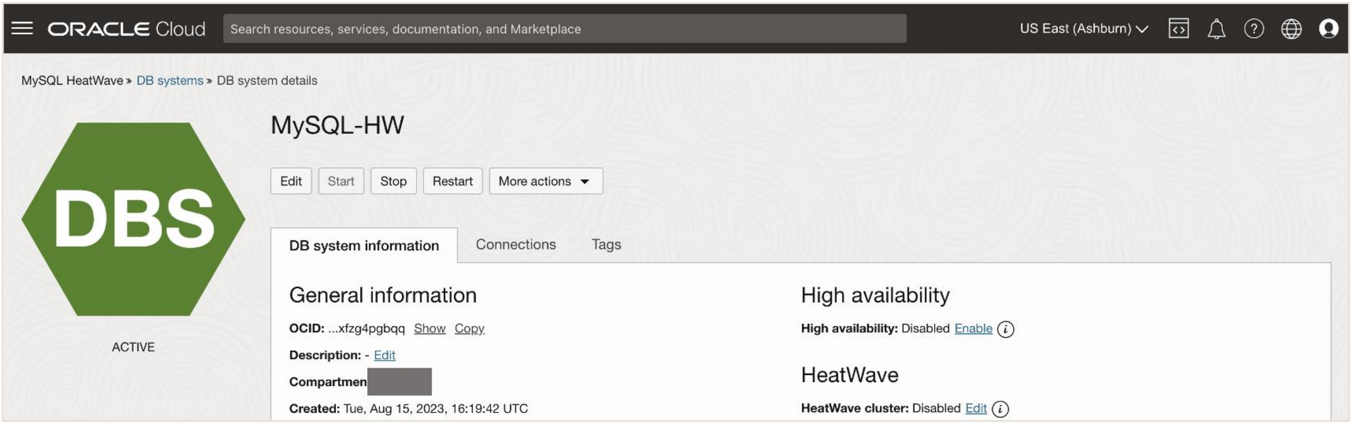
113. After validating, you can have your application/s point to the new HeatWave MySQL instance on OCI.

VII) (Optional) On OCI, if the HeatWave option was enabled during HeatWave MySQL DB creation, add the HW Cluster and load data from MySQL InnoDB storage into the HW Cluster using automation.

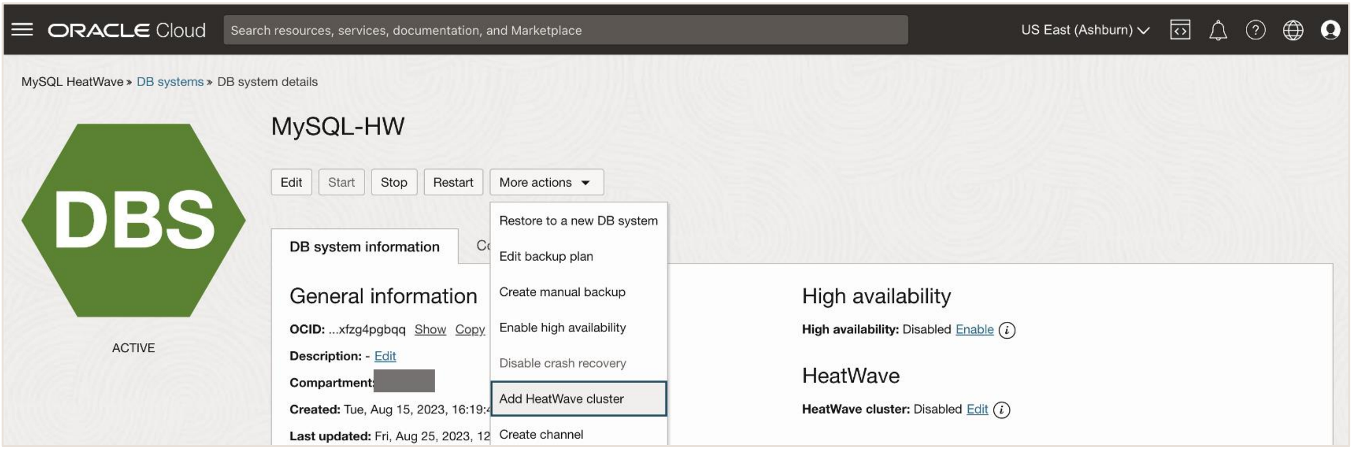
114. Login to [OCI](). Click on the navigation menu, go to **Databases**, and click **HeatWave MySQL**.

115. Click on the name of your HeatWave MySQL instance to go to the **DB System Details** page.

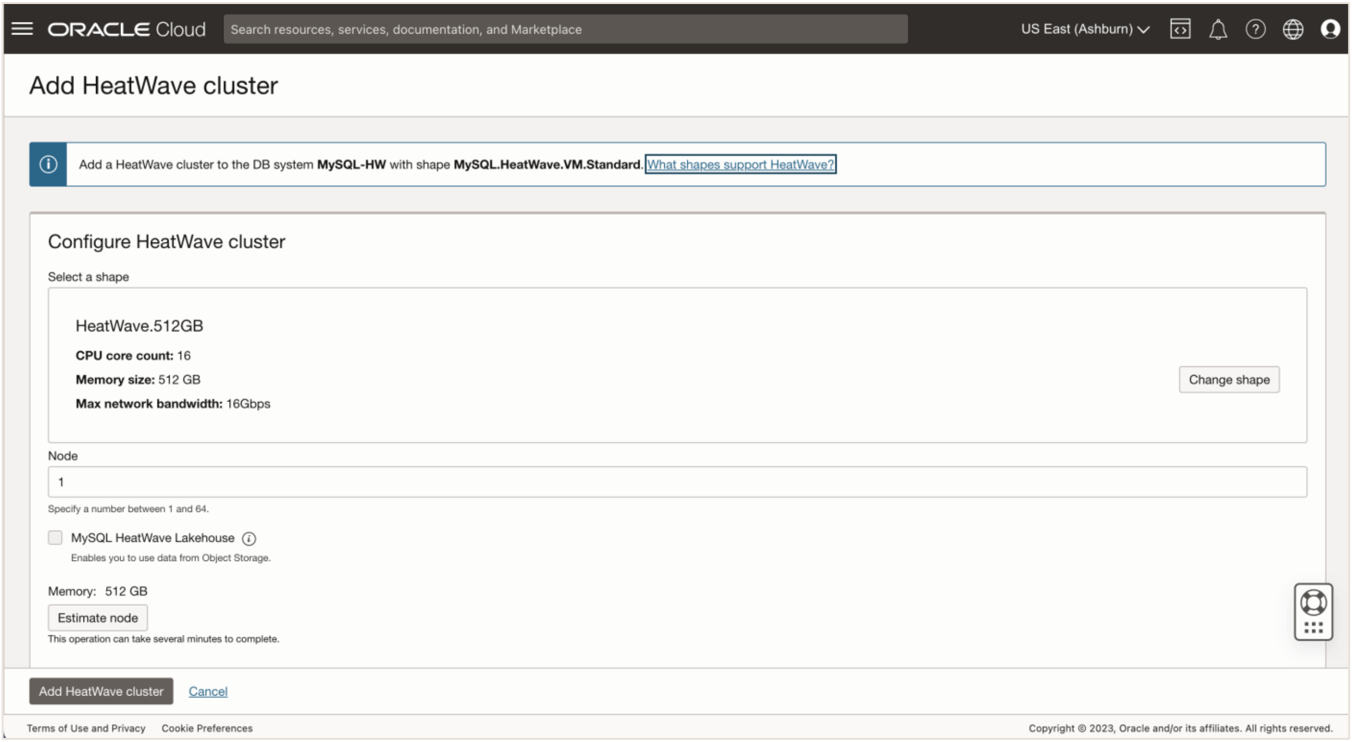




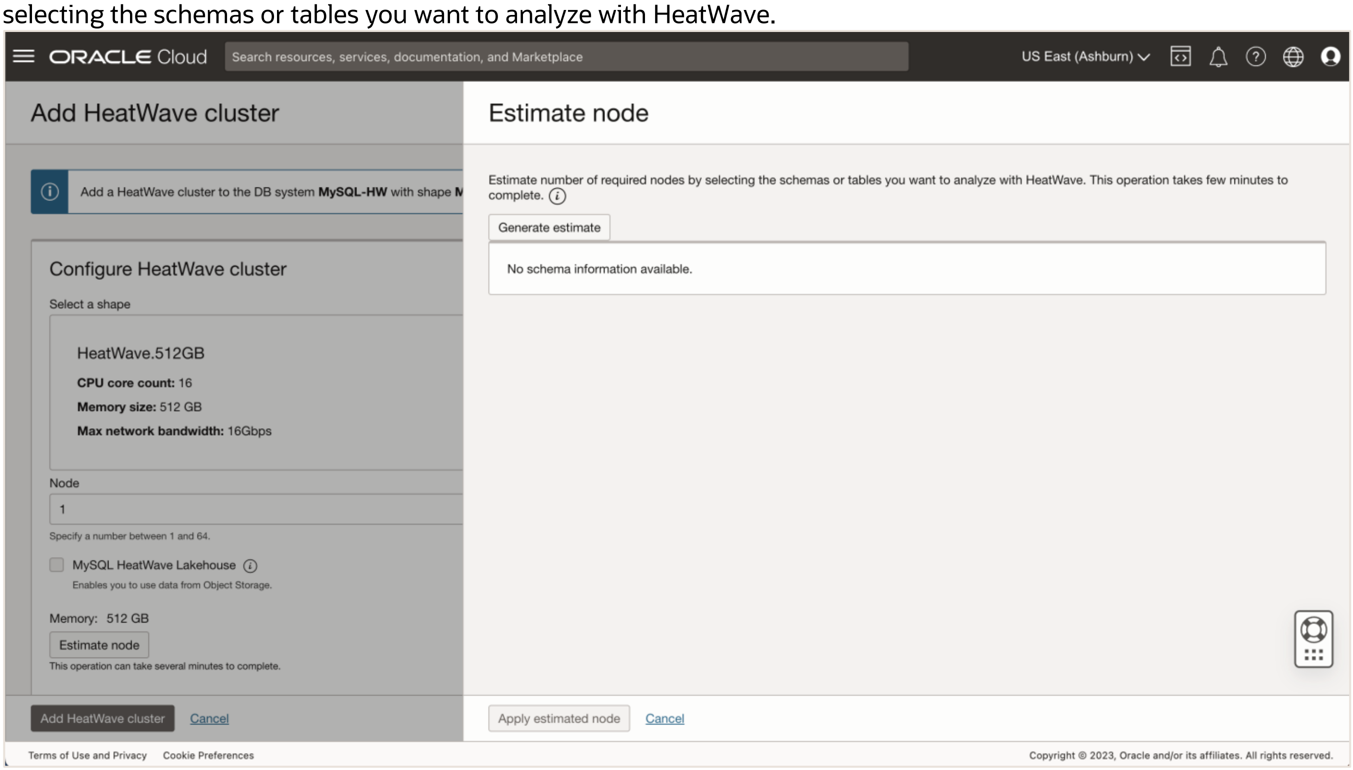
116. Click **More actions** and click **Add HeatWave cluster**.



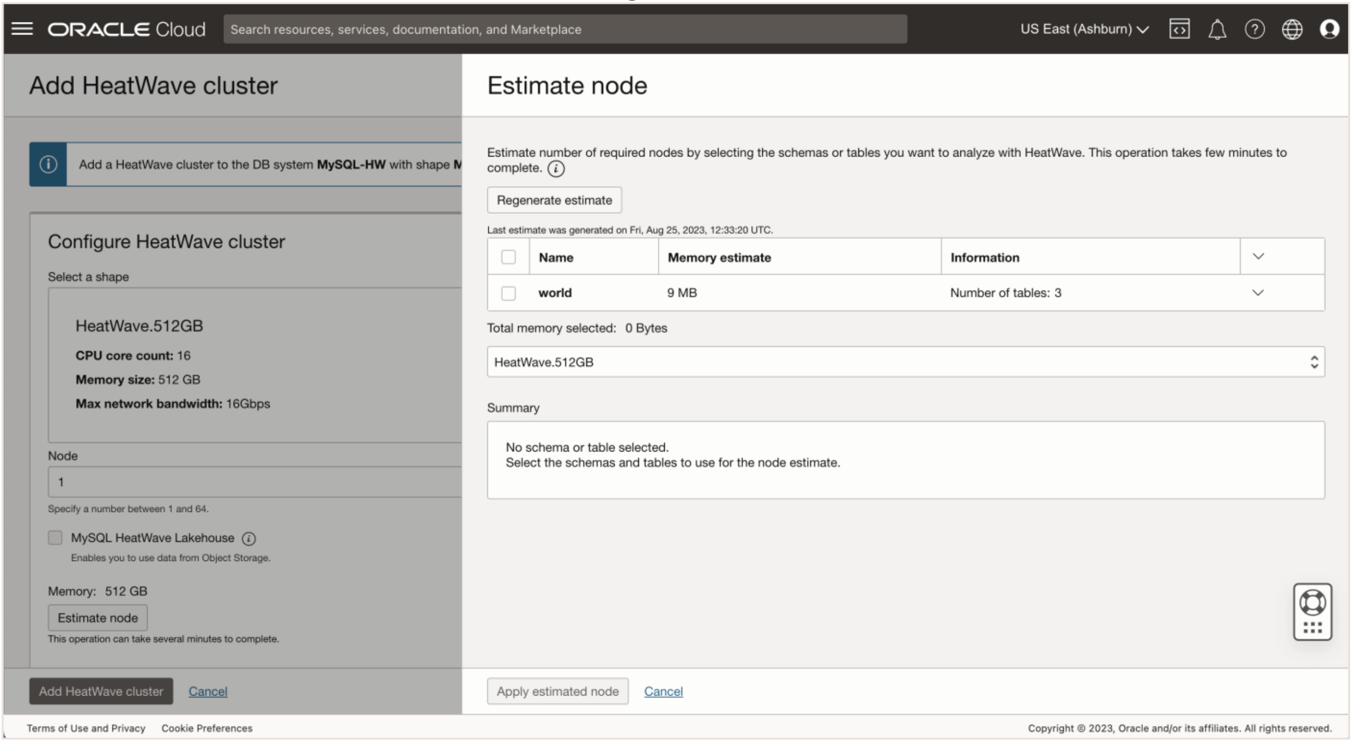
117. Click **Estimate node**.



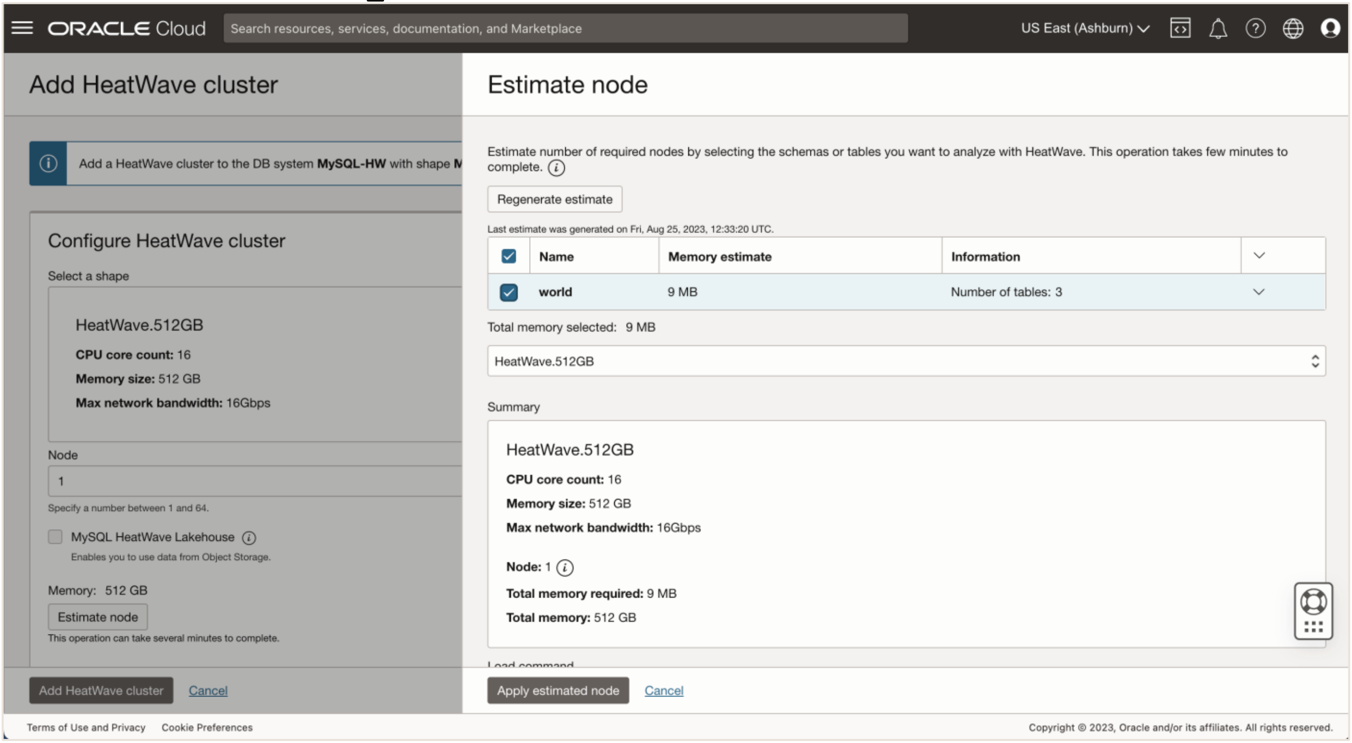
118. Click **Generate estimate**. This step will estimate the number of HeatWave nodes required by selecting the schemas or tables you want to analyze with HeatWave.

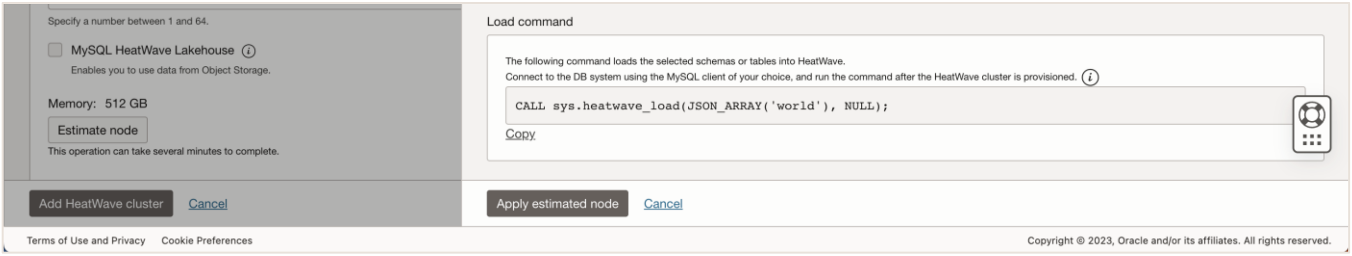


119. Within a few minutes, the list of your schemas that are in the MySQL InnoDB storage engine will be listed. **Check the box** next to the schema or table name that you wish to load in HeatWave for query acceleration and to run OLAP and ML workloads - alongside OLTP.

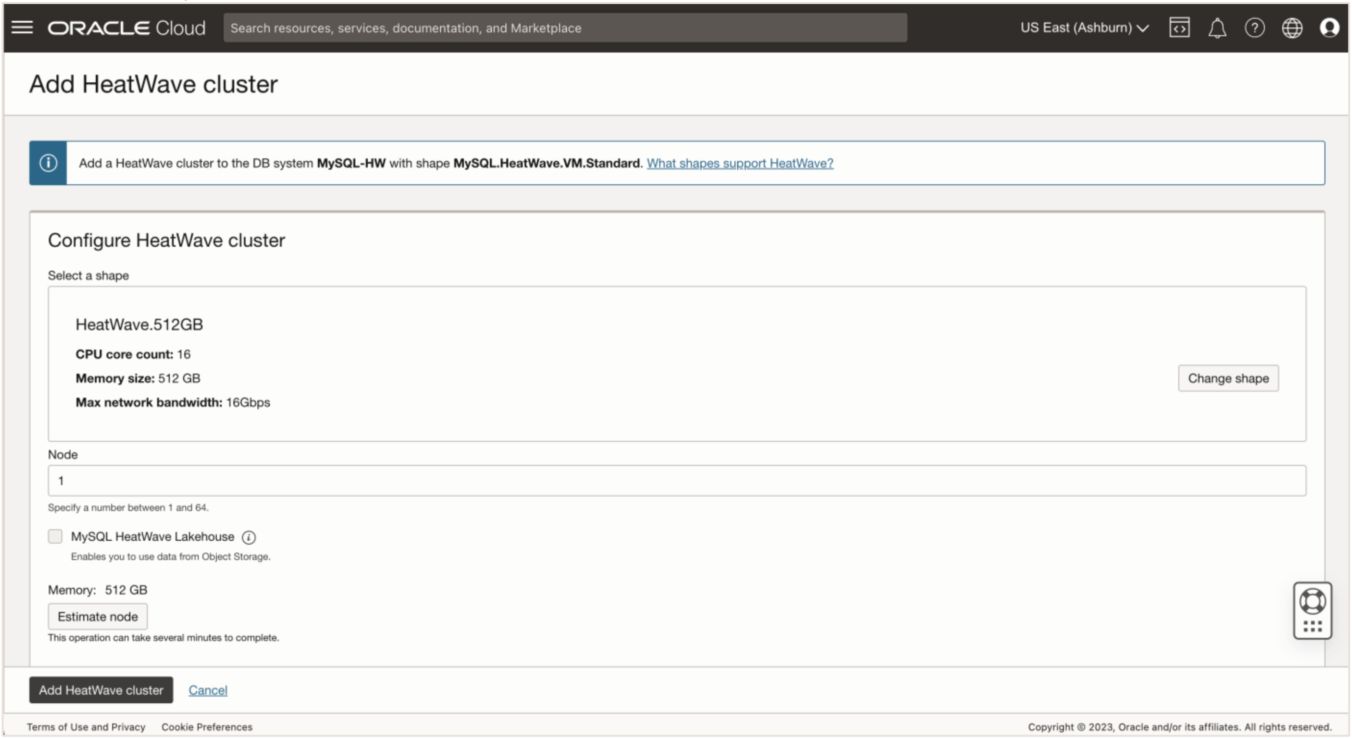


120. After selecting the schemas or tables, scroll down on that page until you see the **Load command**. Copy the `CALL sys.heatwave_load` command and save it. Click **Apply estimated node**.

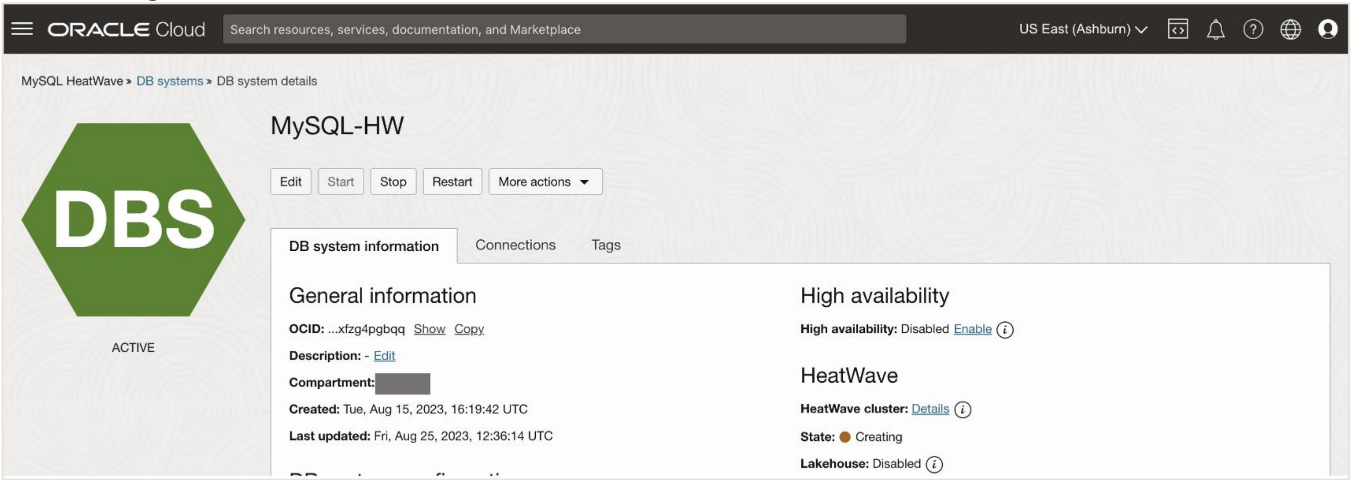


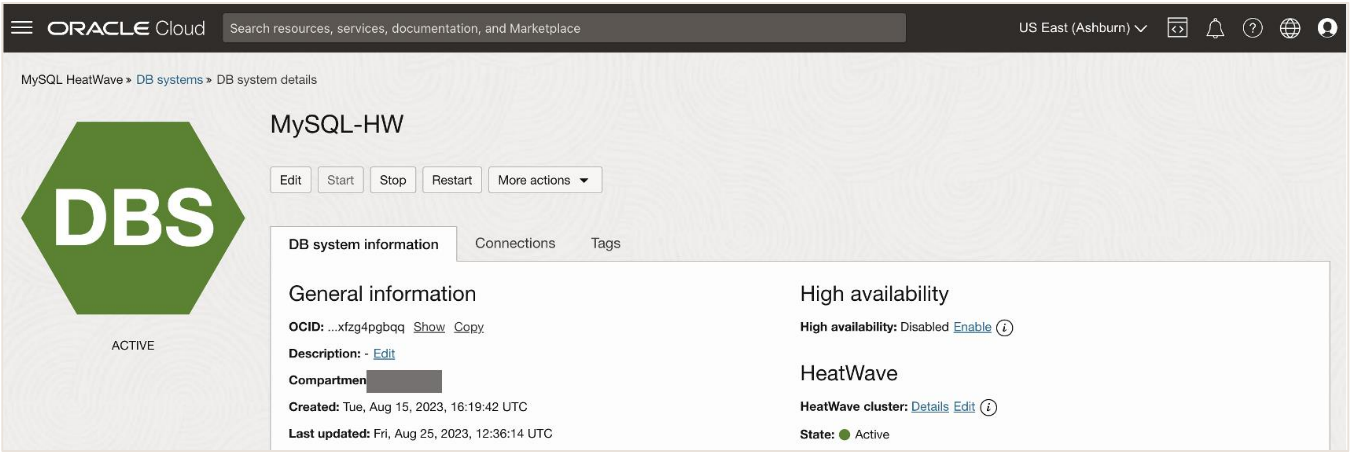


121. Executing the previous step will change the HeatWave node count depending on the data you have selected to load into HeatWave's in-memory engine. Click **Add HeatWave cluster** to finish the HeatWave cluster creation process.



122. The HeatWave cluster will be ready within a few minutes. You should see the HeatWave state change from Creating to **Active**.



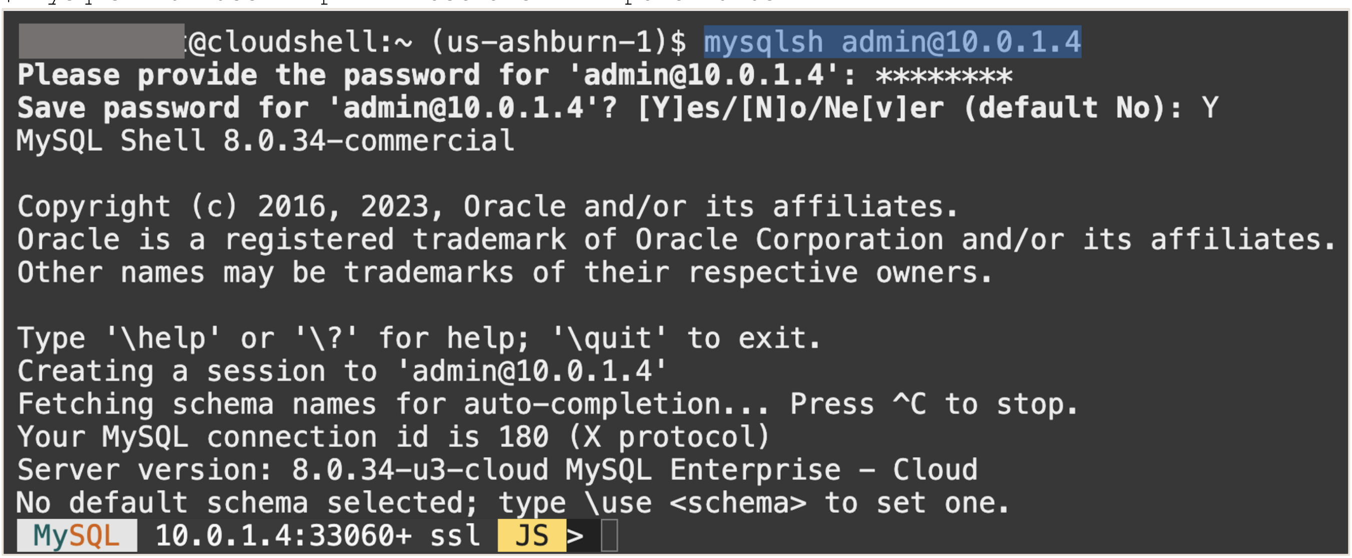


123. Connect to your HeatWave MySQL system using MySQL Shell via Cloud Shell.

```
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```



124. Switch to the SQL mode of MySQL Shell and execute the Load command that we had copied earlier to load data into HeatWave from the MySQL InnoDB storage.

```
MySQL JS> \sql
MySQL SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);
```

Note: replace the `sys.heatwave_load` command with what you have.

```
 MySQL  10.0.1.4:33060+ ssl  world  JS > \sql
Switching to SQL mode... Commands end with ;
 MySQL  10.0.1.4:33060+ ssl  world  SQL > CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);
+------------------------------------------+
| INITIALIZING HEATWAVE AUTO PARALLEL LOAD |
+------------------------------------------+
| Version: 2.20                            |
|                                          |
| Load Mode: normal                        |
| Load Policy: disable_unsupported_columns |
| Output Mode: normal                      |
|                                          |
+------------------------------------------+
6 rows in set (1.2477 sec)

+--------------------------------------------------------------------------+
| OFFLOAD ANALYSIS                                                         |
+--------------------------------------------------------------------------+
| Verifying input schemas: 1                                               |
| User excluded items: 0                                                   |
|                                                                          |
| SCHEMA                       OFFLOADABLE    OFFLOADABLE     SUMMARY OF   |
| NAME                              TABLES        COLUMNS     ISSUES       |
| ------                       -----------    -----------     ----------   |
| `world`                                3             24                  |
|                                                                          |
| Total offloadable schemas: 1                                             |
```

[… output truncated]

```
+-------------------------------------------+
| LOADING TABLE                             |
+-------------------------------------------+
| TABLE (3 of 3): `world`.`countrylanguage` |
| Commands executed successfully: 3 of 3    |
| Warnings encountered: 0                   |
| Table loaded successfully!                |
|   Total columns loaded: 4                 |
|   Table loaded using 1 thread(s)          |
|   Elapsed time: 213.26 ms                 |
|                                           |
+-------------------------------------------+
8 rows in set (1.2477 sec)

+----------------------------------------------------------------------------------+
| LOAD SUMMARY                                                                     |
+----------------------------------------------------------------------------------+
|                                                                                  |
| SCHEMA                          TABLES       TABLES      COLUMNS         LOAD    |
| NAME                            LOADED       FAILED       LOADED     DURATION    |
| ------                          ------       ------      -------     --------    |
| `world`                              3            0           24       1.16 s    |
|                                                                                  |
+----------------------------------------------------------------------------------+
6 rows in set (1.2477 sec)

Query OK, 0 rows affected (1.2477 sec)
 MySQL  10.0.1.4:33060+ ssl  world  SQL >
```

125. You now have a complete HeatWave MySQL cluster.

To learn more about using HeatWave, please visit our documentation.

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle